中国科学: 信息科学 2022 年 第 52 卷 第 1 期: 1–53
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS
评述

面向实时视频流分析的边缘计算技术

杨铮*, 贺骁武, 吴家行, 王需, 赵毅

清华大学软件学院, 北京 100084
* 通信作者. E-mail: yangzheng@tsinghua.edu.cn

收稿日期: 2021–04–18; 修回日期: 2021–07–23; 接受日期: 2021–08–11; 网络出版日期: 2022–01–10

摘要 实时视频流分析在智能监控、智慧城市、自动驾驶等场景中具有重要价值. 然而计算负载高、带宽需求大、延迟要求严等特点使得实时视频流分析难以通过传统的云计算范式进行部署. 近年来兴起的边缘计算范式, 将计算任务从云端下沉到位于网络边缘的终端设备和边缘服务器上, 能够有效解决上述问题. 因此, 许多针对实时视频流分析的边缘计算研究逐渐涌现. 本文首先介绍了智能视频流分析和边缘计算的背景知识, 以及二者结合的典型应用场景; 接着提出了现有系统所关注的衡量指标和面临的挑战; 然后从终端设备层次、协作层次、边缘/云层次对本领域的关键技术分别进行了详细的介绍, 重点涉及了模型压缩和选择、本地缓存、视频帧过滤、任务卸载、网络协议、隐私保护、查询优化、推理加速和边缘缓存技术. 基于对上述各项核心技术的有机整合, 本文提出了基于边缘计算的视频大数据智能分析平台 Argus, 从数据采集、推理分析, 到数据挖掘、日志管理, 对实时视频流分析全生命周期提供支持, 并成功应用到智慧油田中. 最后, 本文讨论了本领域尚待解决的问题和未来研究方向, 希望为今后的研究工作提供有益参考.

关键词 边缘计算, 视频分析, 模型压缩, 任务卸载, 查询优化

1 引言

随着智能手机的普及和监控摄像头的大规模部署, 全球范围内实时生产的视频数据量呈现出指数增长的趋势. 根据思科公司的预测 [1], 到 2023 年, 视频流将占据互联网总流量的 80%. 传统的人工方法难以对如此大规模的数据进行分析, 因此需要利用计算机自动提取出视频流中的关键信息. 实时视频流分析, 指通过计算机视觉算法对一个或多个摄像头产生的视频流内容自动进行分析和理解, 从而在视频流录制和传输的同时完成目标识别、异常检测等复杂任务 [2]. 由于摄像头源源不断地产生视频流, 自动视频分析必须能够实时进行. 自 1970 年至今, 智能视频分析系统就一直是学术和工业界的热点问题. 实时视频流分析可以被应用在智能安防、智慧城市、自动驾驶, 以及个人生活助理等领域, 不仅能替代传统的人工监控业务, 还可以拓展视频分析任务的应用边界. 自 2012 年 AlexNet [3] 提出以

引用格式: 杨铮, 贺骁武, 吴家行, 等. 面向实时视频流分析的边缘计算技术. 中国科学: 信息科学, 2022, 52: 1–53, doi: 10.1360/SSI-2021-0133
Yang Z, He X W, Wu J H, et al. Edge computing technologies for streaming video analytics (in Chinese). Sci Sin Inform, 2022, 52: 1–53, doi: 10.1360/SSI-2021-0133

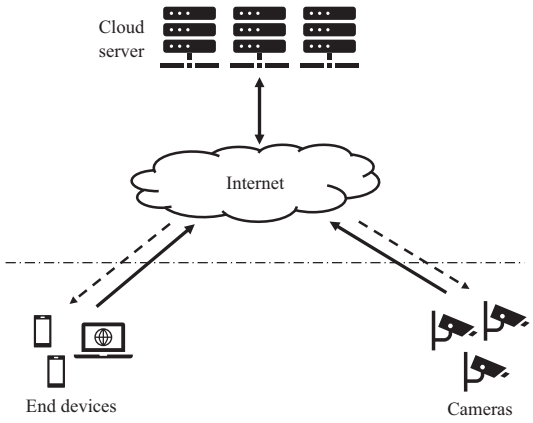

图 1　云计算视频分析架构

Figure 1　Cloud-based video analytics paradigm

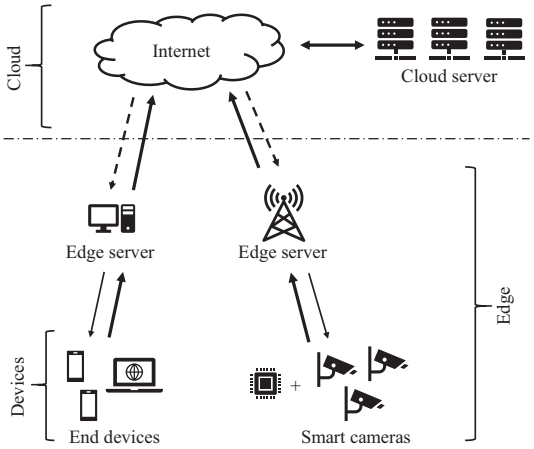

图 2　边缘计算视频分析架构 (参考文献 [7])

Figure 2　Edge-based video analytics paradigm (adapted from [7])

来, 计算机视觉技术, 特别是基于深度卷积神经网络 (convolution neural network, CNN) 的目标检测和识别技术出现了重大突破. 机器在自动视频分析任务上已经可以达到甚至超过人类水平, 使智能视频分析从理论研究快速走向了实际应用 [4, 5]. 深度卷积神经网络需要巨大的计算量, 往往只有在图形处理单元 (graphics processing unit, GPU) 等专用硬件上才能实时运行, 需要将计算任务卸载到具有较强计算能力的设备上. 图 1 为许多传统的实时视频流分析系统所采用的中心化云计算架构: 处于网络边缘的设备端摄像头直接通过互联网将视频传输到网络中心的云服务器. 云服务器获得视频之后对视频进行存储, 并利用 GPU 进行推理分析, 再将推理结果在云端进行整合利用. 在部分场景, 云端还会将最终结果返回给网络边缘的终端设备 [2]. 然而, 基于云计算的视频分析系统面临着带宽占用大、传输延迟高、网络不可靠和隐私保护难等问题.

基于边缘计算的实时视频流分析范式, 可以很好地解决上述问题. 边缘计算旨在将计算任务从位于网络中心的云服务器下沉到与视频源物理接近的边缘服务器或者智能终端设备上 [6]. 如图 2 [7] 所示, 在边缘计算架构中, 计算任务可以卸载到设备 – 边缘 – 云 3 个层级中. 拥有一定计算能力的智能设备和边缘服务器能够在视频源附近直接处理大部分存储和分析任务, 云服务器仅在必要情况下提供计算支持和数据融合. 因此, 基于边缘计算的视频流分析系统占用的互联网上传带宽将显著减少. 此外, 边缘服务器与视频源物理位置接近, 其数据包往返时间 (round trip time, RTT) 相比云服务器几乎可以忽略不计. 基于边缘计算的视频流处理系统只需要考虑视频流的传输时延和视频分析的时延, 能够完成服务时延敏感的增强现实等任务 [8]. 同时, 相比于互联网, 视频源和网络边缘的连接相对简单可控, 能够确保足够的传输带宽和稳定性. 将视频数据在边缘端直接处理, 或者在边缘端进行脱敏化操作之后上传, 也可以有效避免云端隐私泄漏.

除了实时视频分析以外, 许多应用场景也需要对大量的已有视频进行离线分析. 例如, 大型视频网站需要自动分析用户上传的视频, 检测其是否涉黄; 公安执法部门需要在刑事案件发生后回溯已有监控视频, 确定嫌疑人的历史轨迹. 这些对已有视频的离线查询, 需要从数小时甚至数月长度的大量视频中检索出具有特定目标的少量视频帧或片段. 面对后台保存的大规模视频数据, 逐帧进行神经网络分析会耗费大量时间和计算资源. 因此, 离线视频分析的核心问题在于: 如何在不完全分析所有视

频帧的情况下检索出视频帧中的所有目标. 其中最直观的方法就是用底层计算机视觉方法检测视频内容变化的情况, 仅在变化较大时进行分析 [9]. 离线分析的特点是不需要按照视频时间顺序进行, 因而许多工作都在分析前从整个视频中选出部分帧训练定制化的轻量级神经网络模型 [9~13]. 定制化模型能够较好适应已有视频的特性, 可以在精确度损失较少的情况下减少单次分析带来的开销. 在此基础上, NoScope [9] 和 THAHOMA [12] 利用不同轻量级模型组成多级过滤器, 在离线分析过程中筛选掉大部分视频帧不进行完整分析. Focus [10] 则利用轻量级模型在存储视频时构建索引, 处理后续任务时仅对包含索引的部分视频帧用完整模型进行分析. 本文主要是面向实时视频流分析, 不再进一步展开介绍离线视频分析.

2017 年, 美国微软公司的 Ananthanarayanan 等 [14] 提出 "实时视频分析是边缘计算的杀手级应用", 并且分享了一系列针对智能交通分析的边缘实时视频流分析系统 [15~17]. 此后, 许多针对实时视频分析的边缘计算系统和研究开始涌现, 并且逐渐被部署到真实世界的各种应用中. 目前已经存在许多针对实时视频流分析的边缘计算研究工作, 但仍然缺乏系统性的综述工作. 因此, 本文希望对基于边缘计算的实时视频流分析领域的最新进展和未来方向进行讨论.

1.1 相关调研和综述

在边缘计算、深度学习、视频分析领域, 已经存在一些相关调研和综述. 我们对其中一些有代表性的工作进行对比分析.

一部分工作针对智能视频分析. Liu 等 [2] 调研了从 1970 到 2013 年的智能视频分析算法和系统. 他们总结了视频分析系统的软硬件架构、分析目标, 以及相应的计算机视觉算法. 由于发表时间较早, 该工作没有涉及基于深度学习的视频分析算法. 作为机器学习系列丛书的一部分, Olatunji 等 [5] 回顾了智能视频分析系统相关的基础理论和计算机视觉算法, 并分析了他们在交通、增强现实、安防等领域的可能应用. Liu 等和 Olatunji 等都只考虑了将计算任务卸载到摄像头或云端, 没有提及基于边缘计算的分布式视频分析架构.

一部分工作针对边缘计算技术. Mao 等 [18] 对移动边缘计算 (mobile edge computing, MEC) 涉及的通信、计算、设备和服务器模型作出了形式化定义. 他们重点调研了移动边缘计算中的无线电和计算资源协同分配问题, 并提出利用边缘缓存优化视频流直播效果. Li 等 [19] 总结了边缘计算和雾计算中的计算范式和分层模型, 并调研了边缘计算涉及的跨层资源分配算法. 他们提到了视频传输对网络带宽的压力, 并且分析了如何通过多层边缘计算架构减少视频流传输延迟. Mao 等和 Li 等主要考虑了边缘计算架构中的一般性研究, 较少涉及具体的应用场景.

还有一部分工作针对人工智能和边缘计算的结合. Zhou 等 [20] 提出了边缘智能 (edge intelligence, EI) 的概念, 并分别讨论了在网络边缘进行深度学习模型训练和推理的关键技术. Zhou 等指出了实时视频分析是研究边缘智能的重要动机, 并且涉及部分视频帧过滤模型. 与 Zhou 等相似, Chen 和 Ran [7], Murshed 等 [21] 分别调研了深度学习和机器学习在边缘计算系统的应用, 虽然和本文的调研领域有所重叠, 但是并未完整覆盖视频实时分析和传输协议的特性.

最近, 部分工作尝试对视频分析和边缘计算进行更深入的调研. Zhang 等 [22] 调研了边缘计算和视频流分析技术在公安、交通、医疗等公共安全部门的应用. 该工作从各部门实际部署的系统出发, 对涉及的视频分析任务和计算机视觉算法作了简述. Jedari 等 [23] 介绍了互联网视频流传输协议的基础知识, 以及无线和蜂窝网络技术对边缘计算的支持. 他们还分析了无线网络边缘计算在视频流传输和分析中的各种应用. Zhang 等和 Jedari 等的工作都部分包含了边缘视频分析, 但是他们仅按照应用部门对各个系统分别阐述, 对边缘实时视频分析中涉及的共性关键技术没有系统性分析和总结.

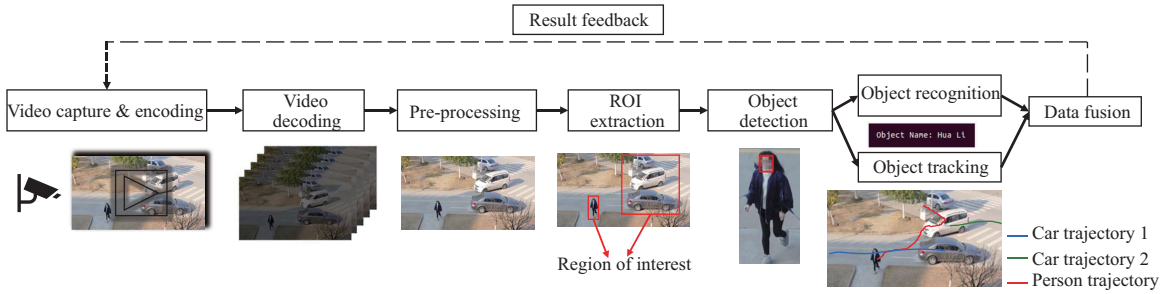

图 3 (网络版彩图) 经典的视频分析流程, 以人脸识别为例

Figure 3 (Color online) Typical video analytics pipeline, taking face recognition as an example

1.2 本文贡献

本文的贡献如下:

- 收集和整理了现有的针对实时视频流分析的边缘计算系统和研究, 并将其中涉及的关键技术分类为 3 个层次: 终端设备层次、协作层次, 和边缘/云层次. 我们分别对 3 个层次中的重要技术的应用和发展进行了总结.
- 提出了基于边缘计算的视频大数据智能分析平台 Argus, 并分别从视频/传感器数据源、接入网络、边缘视频流分析、模块编排和管理、数据挖掘, 以及业务逻辑等方面对其进行了介绍.
- 提出并分析了边缘实时视频流分析当前面临的挑战和未来可能发展方向.

第 2 节介绍了实时视频流分析和边缘技术涉及的背景知识和应用场景. 第 3 节指出了基于边缘计算的实时视频流分析系统的关键指标和面临的挑战. 第 4 节分层次总结边缘实时视频流分析涉及的关键技术. 第 5 节提出了 Argus 平台并阐述其顶层设计. 第 6 节讨论了本领域尚待解决的问题和可能的发展方向. 最后, 第 7 节对本文进行了总结.

2 背景介绍

本节简单介绍背景知识及相关术语, 包括视频分析流程、典型的边缘计算范式, 以及基于边缘计算的实时视频流分析的具体应用场景.

2.1 视频分析流程

实时视频流分析需求较为复杂, 通常涉及多个不同模块的组合. 图 3 以常见的目标识别与追踪为例, 对视频分析所涉及的流程进行了展示. 从逻辑层面上, 实时视频分析流程可以被划分为视频采集和编码、视频解码、预处理、区域提取、目标检测、目标识别、目标追踪, 以及数据融合这几个部分. 在必要时, 系统还需要将分析结果返回给视频源用于实时渲染或决策.

视频采集和编码: 从物理环境中拍摄连续的视频帧, 然后按照 H.264 [24], H.265 [25] 等视频编码协议, 通过软件或硬件方式将视频帧压缩成视频比特流. 利用连续视频帧间的时间和空间冗余, 视频编码协议可以在保持视频质量的前提下减少视频大小. 一方面, 编码协议会对视频帧本身进行帧内压缩. 另一方面, 大部分编码协议还采用了基于块匹配的运动补偿技术减少帧间冗余. 其中, 不同视频帧间匹配的相似图像块的位置变化被称为运动矢量 (motion vector), 图像块的内容差异则被称为残差 (residual).

视频解码: 将接收到的视频流按照帧率、分辨率等参数, 根据视频编码协议进行解码, 从而提取出其中的视频帧. 常见的视频分析算法都采用逐帧分析的模式, 因此视频解码通常是视频流分析中的必要步骤. 与视频编码相对应, 解码器仅需要根据运动矢量和残差对参考帧进行简单计算, 就可以推断出视频流中的其他视频帧.

预处理: 对原始视频流中提取出的视频帧进行图像处理, 使之满足具体分析任务的要求. 例如, 对畸变的视频帧进行去畸变操作, 对图像进行去噪、增强、亮度调整等操作. 有的下游分析算法还需要将视频帧预处理成灰度图、亮度图等特定格式 [26].

区域提取: 从视频帧中提取出可能出现目标的感兴趣区域 (region of interest, ROI). 其中, 监控和交通等领域常见的固定摄像头通常采用背景抠除 [27] 方法, 将当前视频帧减去背景图像来得到视频帧中的动态目标. 手机和无人机上的可移动摄像头, 则可以采用光流法计算出像素的移动, 进而确定 ROI [28].

目标检测: 从 ROI 中确定特定类别的目标是否存在, 并通过数据框形式标记目标位置和大小. 常见的目标包括行人、车辆、人脸等. 传统的目标检测方法先利用 HAAR, SIFT, HOG 等人工方法提取出图像的特征, 再通过 SVM, AdaBoost 等浅层机器学习算法判断图像的所有滑动窗口中是否包含目标 [29]. 深度学习方法则利用了更深的卷积神经网络架构, 可以学习到比传统人工特征更有效的特征. 由于区域提取网络 (region proposal network, RPN) [30]、固定网格回归 (fixed-grid regression) [31] 等方法可以直接预测目标位置, 深度学习还能避免滑动窗口算法带来的计算开销 [32].

目标识别: 在目标检测的基础上试图提取出单个目标的身份信息. 该模块首先通过人工或深度学习方法提取出当前目标的特征, 然而将特征向量在已有数据库中进行比对, 试图判断当前目标的身份 [33]. 许多智能视频分析系统的业务都直接依赖于目标识别的结果. 例如, 智慧零售系统需要通过该模块识别当前客户身份, 从而定制个性化推荐和提供无感支付.

目标追踪: 在多个连续的视频帧之间实时追踪同一目标的轨迹变化. 该方法可以减少重复进行目标检测或识别带来的计算开销. 目标追踪算法可以分成基于检测框重叠关系, 基于 HOG、边缘检测等人工特征, 以及基于卷积神经网络的深度学习特征这三大类方法 [34,35]. 此外, 部分算法还考虑了在目标检测的基础上, 利用匹配算法同时对多个目标进行追踪 [36].

数据融合: 整合多个摄像头的分析结果, 并利用数据库中的历史信息进行数据挖掘. 由于完整的视频分析系统通常涉及多个摄像头组成的网络, 数据融合能得到比单个视频流分析更有价值的信息. 例如, 在大范围区域中定位出某个人或车辆的完整运动轨迹, 利用不同路段的多个交通监控摄像头判断和预测城市车流状况等 [37]. 注意到数据融合不仅仅发生在视频分析流程的末端, 也有可能出现在目标追踪, 甚至区域提取部分.

不同的视频分析任务可能会将不同的视频分析模块进行整合. 例如, 部分增强现实任务需要跳过目标识别, 直接返回目标检测的结果从而进行实时渲染; 自动驾驶任务不仅需要目标的检测框, 还需要通过图像分割确定每个目标的像素级位置. 同时, 基于深度学习的计算机视觉技术, 可以直接对上述一个或者多个部分直接进行端到端的学习. 例如, YOLO [31], SSD [38] 等目标检测模型不需要区域提取, 可以直接从整张图像直接推理出目标的位置和类型. Zhu 等 [39], Feichtenhofer 等 [40] 提出的视频分析模型甚至可以直接处理连续多帧视频, 在检测出目标位置的同时完成目标追踪. 因此, 上述视频分析流程仅作为逻辑划分, 每一个模块不一定能在实际的视频分析系统中一一对应.

如果读者对视频分析的基础理论和各个模块的计算机视觉算法感兴趣, 可以参考 Olatunji 等 [5] 的工作.

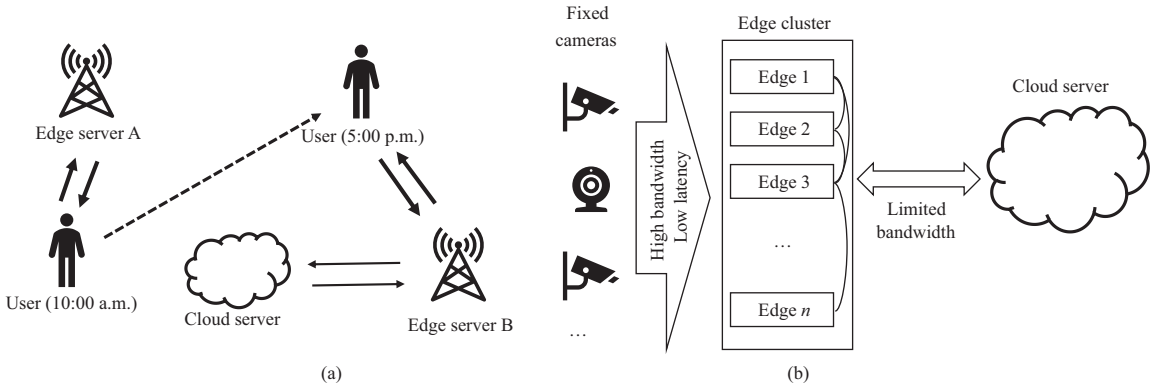

图 4　常见边缘计算范式. (a) 游牧服务模式; (b) 边缘站点模式

Figure 4　Typical edge computing paradigms. (a) Cyber-foraging mode; (b) edge site mode

2.2　边缘计算架构

边缘计算是近年的研究热点, 有许多与其相关的概念, 例如雾计算、移动边缘计算、微型云计算 (cloudlet computing) 等 [41]. 我们认为, 这些概念都属于边缘计算的范畴, 即将计算负载分流到数据生产源附近的网络边缘端. 如图 2 所示, 定义中的网络边缘端, 既包括位于基站中、工厂内的边缘服务器, 也包括具有一定计算能力的智能手机、摄像头等终端设备.

根据物理设备的算力和与数据源的距离, 可以进一步将边缘计算系统中的设备纵向分为 3 层: 底层是计算能力较弱、作为数据源的终端设备, 中层是计算能力相对较强但有限的边缘计算服务器, 顶层是具有弹性扩张能力的云服务器. 设备端和边缘服务器之间具有稳定的网络连接, 同时具备带宽大和延迟低的特点; 而边缘服务器和云服务器之间则通过因特网进行连接, 具有连接不稳定、带宽低、延迟高等特点. 按照计算负载在 3 类设备上的分配, 常见的边缘计算系统可以纵向分为设备 – 边缘、边缘 – 云端, 以及设备 – 边缘 – 云端 3 种架构. 同时, 同一类设备之间也可以相互横向卸载计算任务, 从而组合出更复杂的多设备协同架构 [19].

边缘计算最初的设想是为移动设备提供游牧服务 (cyber-foraging), 如图 4(a) 所示: 当手机或可穿戴设备需要进行计算量较大的人机交互任务时, 会搜索附近的边缘服务器, 提交一次性的低延迟分析请求, 并接收计算结果. 移动设备可能会在一次任务中随着自身位置变化向多个边缘服务器先后发出请求, 一个边缘服务器可能同时服务多个租户的不同任务 [42]. 游牧服务更强调边缘计算所带来的低延迟响应, 常见于增强现实等交互应用, 多采用设备 – 边缘 和设备 – 边缘 – 云端架构.

随着边缘计算在工业场景的应用, 如图 4(b) 所示的边缘站点 (edge site) 模式逐渐兴起: 在工厂、十字路口等地点部署固定摄像头和边缘计算服务器集群, 7 × 24 小时稳定地对大量数据进行持续分析和存储 [43]. 由于边缘服务器与摄像头位置相近, 网络条件较为简单, 二者间可以保持稳定的高带宽和低延迟通信. 边缘站点不涉及多租户服务, 强调服务的稳定运行和节约云边通信带宽, 常见于安防监控、工业分析等场景, 并且多采用边缘 – 云端和多设备协同架构.

虽然应用场景不同, 但是二者殊途同归, 最终都是为了解决终端设备计算能力不足的问题, 并且在分析精度、端到端延迟、资源占用等多方面进行权衡. 区别在于二者面临的应用场景不同, 因此对延迟、精度、带宽等参数的容忍度不同. 例如, 游牧服务模式的增强现实任务可能需要在 20 ms 内传回结果, 边缘站点上运行的油井状态检测可能可以接受 10 s 以内的延迟 [43].

2.3 应用场景

如 2.2 小节中介绍, 边缘计算的服务模式可以大致分为游牧服务和边缘站点两类. 在边缘实时视频分析的众多应用场景中, 增强现实、可穿戴认知辅助、无人机搜救、自动驾驶等任务常利用游牧服务模式; 而安防监控、智慧零售、交通分析等应用则更多采用固定的边缘站点模式. 接下来, 我们对几个典型的应用场景进行简介.

2.3.1 安防监控

智能安防监控是实时视频分析部署最广泛的应用场景之一. 据报道, 全球范围内安装了超过十亿个监控摄像头[1)]. 智能安防监控利用大量的固定摄像头组成网络, 实时检测非法行为、追踪犯罪人员, 从而维护公共安全. 例如, 中国的天网计划将全国不同地点的监控摄像头联网, 并可以利用人脸识别在数秒内定位目标罪犯; 美国的安铂报警 (AMBER alert) 系统, 利用摄像头网络对幼儿绑架车辆进行追踪, 并实时广播警报. 这之中主要涉及人脸识别、行人和车辆重识别等计算机视觉技术.

利用边缘计算, 可以在智能摄像头及摄像头附近对视频进行提前分析, 减少视频传输到中央云服务器的带宽压力, 提升一个区域内同时支持的智能摄像头路数和任务吞吐量. Zhang 等 [44] 提出了 Vigil, 通过在终端设备上进行多个摄像头间的视频帧筛选和网络整流, 有效降低了带宽占用, 能够比直接上传视频流多覆盖 5 到 200 倍的监控区域. 根据 Hu 等 [45] 的实验, 仅仅将视频分析和人脸特征提取放在边缘服务器上, 让服务器根据特征进行人脸匹配, 就可以节省 90% 的网络带宽, 从而缓解了大规模人脸识别的带宽瓶颈. Zhang 等 [46] 则调动多个本地边缘设备协作进行实时车辆追踪, 减小了安铂警报的延迟.

2.3.2 交通分析

交通拥堵是全球所有大城市面临的共同问题, 对交通情况进行实时监控, 可以辅助交通部门进行城市车流规划, 缓解交通拥堵压力. 同时, 通过对交通事故进行监测, 可以及时部署救援队伍, 减少事故死亡率. 与安防监控类似, 交通分析同样利用了大量固定摄像头组成的网络进行分析, 主要利用了目标检测、多目标跨摄像头追踪等技术.

Ananthanarayanan 等 [14] 在美国的 Bellevue 等城市的十字路口部署了边缘实时视频分析, 对通过路口的车辆数量和方向进行计数, 从而智能控制交通信号灯的时长. 经过视频分析流程的优化, 每个十字路口的分析系统都可以由一个边缘服务器进行服务, 并且保持 95% 以上的分析准确率, 减少了带宽占用和部署成本. 与之相似, Barthélemy 等 [47] 在澳大利亚的 Liverpool 市部署了基于边缘计算的实时多模态交通分析系统, 用于指导城市建设. 他们用 15 个 Jetson TX2 开发板组成边缘服务器, 能够同时服务 20 路视频流, 并且按照 20 帧每秒 (frame per second, fps) 的处理速度进行跨摄像头车辆追踪和分析. Kar 等 [48] 则另辟蹊径, 利用在汽车上的小型开发板 (Jetson TX1) 上运行的压缩模型对行车记录仪进行实时分析, 试图在没有交通摄像头的街道分析汽车流量和速度. Qiu 等 [49] 提出了 Kestrel, 在云端分析固定摄像头组成的网络同时追踪多个车辆的路径, 并将可能路径发送给边缘端的智能手机, 用手机拍摄到的视频进行轻量级路径筛选. 在仅消耗少量带宽和能量的情况下, 边缘计算将车辆路径预测的精度从 65% 提升到了 97.7%.

1) The top 10 most surveilled cities in the world. 2020. https://www.usnews.com/news/cities/articles/2020-08-14/the-top-10-most-surveilled-cities-in-the-world.

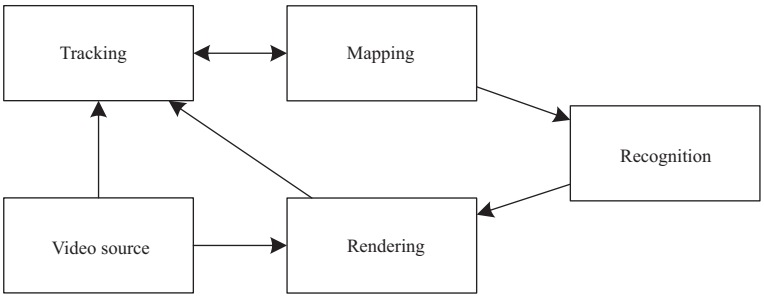

图 5　增强现实应用的视频分析流程图, 其中视频源和渲染部分必须在本地设备上

Figure 5　Video analytics pipeline for AR applications, where the video source and rendering parts must be on the end devices

2.3.3　增强现实

增强现实 (augmented reality, AR) 以及混合现实系统, 基于对视频环境的理解, 在用户视野上渲染虚拟叠加层, 旨在提供真实和虚拟世界融合的沉浸式体验. 为了更好地与真实世界进行交互, 增强现实系统需要能够准确识别出周围的目标位置, 涉及目标检测以及实例分割算法. 例如, 通过识别和标记周围的行人提醒汽车驾驶员危险情况, 或者在人脸上添加跟随表情变化的特效. 图 5 为常见的增强现实分析流程示意图. 系统需要实时对目标进行识别与追踪, 并收集周围环境信息构建出三维地图. 不同于经典的视频分析流程, 增强现实需要把目标检测和识别的结果与当前帧一起渲染到设备端, 并满足人眼对视频流畅性的要求. 因此, 增强现实对视频分析的结果实时性要求较高, 通常需要在 30 ms 甚至 20 ms 以内刷新结果.

边缘计算架构在视频源附近进行处理的特性, 可以有效降低视频分析的延迟, 但仍然需要其他技术的辅助才能达到实时. OverLay [50] 将增强现实过程中的图像匹配和识别任务卸载到服务器, 实现了有 180 ms 响应延迟的简单增强现实导游应用. Al-Shuwaili 等 [51] 则考虑多人合作增强现实场景, 通过优化问题调整各个用户的负载分流策略, 从而在维持最低帧率的前提下减少每个设备的能量开销. Ran 等 [52] 的 DeepDecision 利用优化问题权衡分析准确度和帧率, 求解最佳的模型, 处理流程和视频压缩率. EAAR [8] 借鉴了边端协同的连续视觉分析系统 Glimpse [53], 在其基础上考虑了视频渲染的时间, 并利用流水线并行技术和 ROI 编码技术进一步降低延迟, 将视频图像识别速度从 30 fps 提升到了 60 fps.

2.3.4　无人机搜救

小型民用无人机可以方便快速地部署到野外进行紧急搜救和山火检测等任务, 也可以在人群中进行罪犯搜索, 具有覆盖范围广、灵活机动等特点. 无人机侦察任务通常涉及多个无人机组成的集群, 在一片范围内进行目标的搜索或者异常事件巡逻, 因此主要涉及目标检测算法和集群协作算法. 图 6 为针对无人机侦察的视频分析系统常见框架. 不同于手机和汽车上的移动分析, 无人机除了进行视频分析, 还需要根据指令或者内置策略规划其飞行路径, 而且飞行距离又和其能耗息息相关.

由于无人机搜救通常在野外较为复杂的场景进行, 其和云端的通信质量受到严重限制, 只有位于附近的边缘计算服务器能和无人机进行稳定的通信. Merino 等 [54] 提出基于无人机集群的山火预测系统. 每个无人机分别拍摄并处理图片, 边缘服务器利用概率模型综合无人机的分析结果, 并预测是否可能发生山火. 他们还在真实世界中放火以测试分析效果. Motlagh 等 [55] 利用无人机进行人群中的目标识别. 相比于借助无人机直接进行分析, 将识别任务分流给基站旁的边缘服务器可以减少 100 倍的

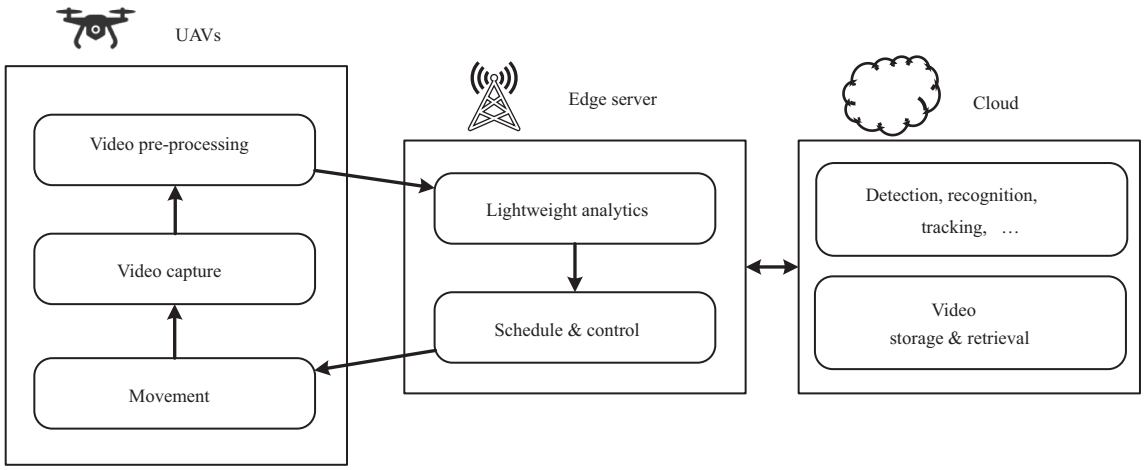

图 6 常见的无人机侦察系统架构

Figure 6 System architecture for drone reconnaissance

能量开销和处理延迟. Kalatzis 等 [56] 基于无人机 – 边缘 – 云端协同进行山火检测, 并利用优化算法在有限的资源限制下尽量减少检测的延迟. 实验表明云边端结合的方案可以在带宽、响应时间、能耗三者之间取得较好的平衡.

2.3.5 可穿戴认知辅助

许多人面临着不同程度的认知障碍, 难以记住人名、识别物体, 或者完成其他日常事项, 例如阿尔茨海默病和脑部外伤患者. 可穿戴认知辅助 (wearable cognitive assistance, WCA), 旨在通过智能眼镜等可穿戴设备, 对佩戴者的第一人称视角视频进行分析, 从而对患者的日常生活和认知行为进行辅助. 例如, 辅助系统可以在摄像头注视人脸时用语音轻声提示人名, 在识别文字后进行朗读, 或在检测到烧水壶等设备后在屏幕上渲染出分步骤操作示范. 其中涉及目标识别、字符识别等计算机视觉任务. 可穿戴认知辅助涉及与人的交互, 因此需要 1 s 以内的延迟.

由于可穿戴设备的大小限制, 其计算能力往往较弱. 而且这些设备通常涉及个人敏感信息. 因此, 需要通过边缘服务器协助可穿戴设备完成复杂计算. He 等 [57] 提出 Gabriel 系统, 针对离线、只有云服务器、有边缘和云服务器 3 种场景设计了 WCA 任务的负载分流方法. 为了尽可能复用各项感知分析技术, Gabriel 将不同的认知辅助任务分别封装到虚拟机中, 并将其部署到了边缘/云服务器上. Zhao 等 [58] 通过可穿戴认知辅助系统自动指导新手使用自动体外心脏除颤器进行急救, 检测人体特定部位并指导使用者进行操作. 实验表明, 如果将任务分流到云服务器而不是边缘服务器, 造成的指令延迟可能会使使用者操作失败. Wang 等 [59] 对 Gabriel 进行扩展, 考虑一个边缘服务器服务多个认知辅助设备的情况. 他们根据用户提供的服务质量 (quality of service, QoS) 函数, 调节 CPU 和内存资源的分配, 最多能同时支持 25 个设备.

3 目标与挑战

本节对衡量视频流分析表现的重要指标进行介绍, 并分析了现有针对实时视频流分析的边缘计算系统需要解决的三大挑战.

3.1 关键指标

我们对大部分边缘实时视频分析系统所关心的关键指标进行一一说明.

延迟: 也称为端到端时延、响应时间, 指从视频帧解码、数据传输、模型推理, 到最终获得分析结果的总时间. 对于需要在本地展示分析结果的应用, 端到端时延还包括结果传回本地以及本地渲染结果的时间. 有一些任务习惯用帧率、刷新率等指标对延迟作出限制, 最大延迟限制就是帧率的倒数. 例如增强现实任务需要的 60 fps 刷新率就对应了 20 ms 以内的端到端延迟. 端到端时延受到视频分析流程中许多因素的影响, 包括数据处理延迟、网络传输方式、设备计算能力、模型参数规模等. 除了完成单次视频分析的平均延迟, 部分系统还关注多任务同时分析时的延迟波动性, 以及延迟的 99 百分位等长尾值.

分析准确度: 是视频分析任务的关键指标, 衡量了视频分析算法和模型的预测准确度. 因此, 针对不同的视觉任务往往对应不同的准确度指标. 对于图像分类和目标识别, 准确度一般指所有预测结果中预测正确的概率. 对其中样本不均衡的情况, 还会用到精度 (precision)、召回率 (recall) 和 F1 分数. 对于目标检测任务, 首先需要根据重叠度 (intersection over union, IOU) 超过阈值判断每个框检测是否成功, 再以此计算准确度、精度、mAP (mean average precision) 等评价指标. 而实例分割和目标分割等像素级别的任务通常按照像素准确度或各类别 IOU 均值作为评价指标. 除了受模型本身的影响, 分析准确度也和运行时间有关. 如果分析结果出现的速率小于视频的帧率, 可能会出现结果与当前视频帧不匹配的现象, 降低分析准确度.

带宽: 指终端设备和边缘服务器、边缘服务器和云服务器之间每秒传输的数据量大小. 在基于移动蜂窝网络通信的系统中, 通信带宽费用可以占据系统的大部分运行成本, 而且网络状况面临波动的可能. 在私有或者有线的固定带宽网络中通信时, 单个视频流的带宽开销大小决定了整个系统的扩展能力.

吞吐量: 指单位计算资源下, 边缘实时视频分析系统能同时服务的设备数量; 或单位时间下, 同一系统能同时处理的分析请求个数. 吞吐量代表了系统能服务的规模和对计算资源的利用效率. 吞吐量越大, 说明系统在单位资源下能够服务更大的规模. 这一指标既受到用户对准确率、延迟要求的限制, 又与整个系统的资源分配、任务调度、计算和带宽资源相关.

能耗: 指边缘视频分析系统中的各个模块, 在单位时间或单次分析任务内消耗的能量. 能耗指标对手机、无人机等基于电池的移动终端设备有重要意义. 其主要受到计算机视觉模型运行和设备间网络通信的影响, 也和同一设备上 CPU, GPU 等不同处理器的使用相关.

内存占用: 指计算机视觉模型在设备上占用的内存大小. 由于深度学习模型通常具有千万甚至上亿的参数量, 需要足够的内存空间进行加载. 一方面, 移动终端设备仅具有有限的内存资源, 限制了深度学习模型的大小. 另一方面, 边缘/云服务器需要服务多个应用, 模型内存占用也会影响其服务规模. 内存占用主要和计算机视觉模型的参数大小以及设备上的资源分配方式有关.

3.2 主要挑战

基于边缘计算的实时视频流分析具有广泛的应用场景, 对各项关键指标要求各不相同, 并且分别运行在设备 – 边缘 – 云端组合出的不同架构上. 虽然如此, 针对不同应用的边缘实时视频流分析系统, 通常采取相似的技术手段去解决遇到的一些共同问题. 例如, Kar 等 [48] 的车流分析系统和用于认知辅助的 Gabriel 系统 [57] 都利用了模型压缩技术在终端设备上定制模型; 视频监控系统 Vigil [44] 和 Al-Shuwaili 等 [51] 的增强现实系统都利用了边缘设备间的点对点通信优化系统调度.

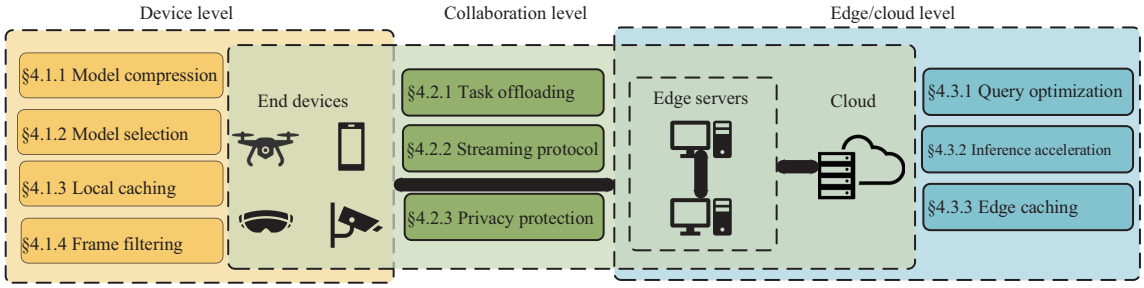

图 7 关键技术层次示意图

Figure 7 Taxonomy of key technologies

我们认为边缘实时视频流分析系统, 主要需要解决以下三大挑战:

第一, 如何将复杂的计算机视觉算法和模块应用到资源有限的终端设备上. 作为视频源的终端设备往往仅具有有限的计算和能量资源, 而且不同设备间硬件架构各不相同. 例如, 搭载 Jetson TX2 的智能设备可以按照每秒 1 帧的速度运行大部分常见的目标检测网络, 而普通的监控摄像头的内存可能根本无法加载正常的神经网络模型. 然而, 终端设备由于离视频源最近, 在低延迟快速分析和视频预处理方面具有无法替代的优势.

第二, 如何高效进行终端 – 边缘服务器 – 云服务器之间的各个设备的任务卸载和协作. 边缘实时视频分析是典型的分布式异构计算场景, 涉及设备 – 边缘、边缘 – 云端、设备 – 边缘 – 云端等架构中的不同层级的负载分配和通信协作, 以及同一层级的多设备协同. 此外, 设备间的通信还面临复杂的网络环境, 需要考虑到对带宽和延迟波动的自动适应.

第三, 如何提高边缘/云服务器在单位资源下的服务规模. 在大规模的视频分析系统中, 边缘/云服务器通常需要每秒处理几十个视频流的上千个分析请求, 并且可能涉及多个 GPU, FPGA 等专用硬件的资源分配. 边缘服务器虽然算力远大于终端设备, 但是也无法弹性扩展, 而云服务器的弹性计算能力面临高昂的经济成本. 因此, 系统需要利用有限的计算能力同时服务尽量多的请求数量.

4 关键技术

第 3.2 小节中提到边缘实时视频流分析系统主要需要解决三大挑战. 因此, 我们针对这些挑战, 分别调研了如图 7 所示 3 个层次的关键技术.

- 终端设备层次: 应用于算力较低终端设备和智能边缘计算节点上的技术. 从模型定制和任务设计的角度出发, 在满足其资源限制的前提下, 在视频源附近完成快速的初始视频分析.
- 协作层次: 应用于云边端不同层级和同层级中不同设备间相互协作的技术. 涉及任务的卸载方式和设备间的协作机制, 在复杂网络条件下综合考虑分析准确度、延迟和能耗, 确保整个视频分析系统的端到端表现.
- 边缘/云层次: 应用于算力较强边缘和云服务器的技术. 对多路视频的大量分析请求进行合理的任务调度、资源分配和冗余去除, 在满足准确度与延迟需求的前提下, 增加系统在单位计算资源下的吞吐量.

注意到, 一个完整的边缘视频分析系统可能会涉及其中一个或者多个层次的不同技术的组合. 其中每个技术最终都是为了按照用户的需求, 分别从视频分析精度、端到端延迟、带宽利用、资源开销等角度优化整个分析任务. 以典型的基于设备 – 边缘 – 云端三级架构的大规模视频监控场景为例, 系

表 1　关键技术相关文献

Table 1　Related work of key technologies

Technology level	Key technology	Sub technology	Refs.
Device level	Model compression	Pruning and quantization	[3, 60∼64]
		Matrix decomposition	[65∼68]
		Architecture redesign	[69∼72]
	Model selection	–	[65, 71, 73∼76]
	Local caching	Result caching	[8, 39, 53, 66, 77∼82]
		Intermediate value caching	[68, 83, 84]
	Frame filtering	Change-based filtering	[53, 85∼87]
		Analytics-based filtering	[59, 88∼91]
		Multi-camera filtering	[16, 44, 92, 93]
Collaboration level	Task offloading	Vertical offloading	[17, 52, 88, 94∼111]
		Horizontal offloading	[63, 112∼119]
	Streaming protocol	Video streaming	[8, 44, 77, 120∼126]
		Feature map streaming	[127∼131]
	Privacy protection	Target denaturing	[132∼136]
		Global transformation	[137∼139]
Edge/cloud level	Query optimization	–	[15, 17, 140∼143]
	Inference acceleration	Batch processing	[144∼152]
		Model merging	[153∼155]
	Edge caching	–	[156∼161]

统需要先通过跨层协作技术, 根据网络状况、任务难度在不同层级进行任务卸载. 而其中卸载到设备端的任务需要利用终端设备层技术粗略分析出视频中的有用信息. 卸载到边缘和云端的任务则需要通过边缘/云服务层技术提高资源利用效率. 本文希望对每个层次中的不同技术的内容和发展进行总结, 为读者梳理边缘实时视频流分析领域的研究脉络.

表 1 [3, 8, 15∼17, 39, 44, 52, 53, 59∼161] 对不同层次的关键技术与相关工作进行了汇总.

4.1　终端设备层次

终端设备层次包括了模型压缩、模型选择、本地缓存, 以及视频帧过滤等技术.

4.1.1　模型压缩

基于深度学习的计算机视觉模型通常由百万甚至上亿的参数组成, 虽然分析准确度很高, 但是对计算和内存资源要求较高, 难以在终端设备上运行. 模型压缩技术通过一系列手段, 在尽量少牺牲分析准确度的情况下减小模型的大小, 降低参数量和计算复杂度, 使之能加载到设备端上运行. 一方面, 降低的参数量可以减小模型的内存占用; 另一方面, 计算复杂度减少也能带来处理速度的提升和能耗的减小. 常见的模型压缩技术包括参数剪枝、模型量化、矩阵分解和模型重构等.

剪枝和量化.　模型剪枝通过将模型中的大部分冗余参数置为零使模型稀疏化, 从而降低模型的内存占用. 模型剪枝通常包含 3 个步骤: 第 1 步, 训练一个初始的神经网络; 第 2 步, 衡量网络中参

数的重要性, 去除冗余的参数; 第 3 步, 固定去除的参数为 0, 微调模型的剩余参数, 避免精度损失. 模型量化是对模型剪枝的补充, 将模型参数的比特宽度进行缩减, 然后重新进行训练. 例如将 32 位的双精度浮点参数变成 16 位的单精度浮点数或者 8 位的整型. Han 等 [60] 按照模型的参数绝对值大小进行剪枝, 并迭代进行第 2 步剪枝和第 3 步微调. 他们在 AlexNet 和 VGGNet 上分别减少了 90.9% 和 87.5% 的参数量, 以及 3 倍和 5 倍的浮点运算数, 并且保持了和原有模型相同的精度. 他们的改进工作 DeepCompression [61] 结合了模型剪枝、模型量化和 Huffman 编码, 将 AlexNet 和 VGGNet 的参数量进一步压缩到 3.05% 和 2.55%. Yang 等 [62] 指出单纯的参数减少不能代表计算能耗的降低, 因为神经网络中大部分参数在全连接层, 而大部分计算操作在卷积层. 他们直接对卷积神经网络的能量开销进行估计, 并以此为指导进行逐层模型剪枝, 在 AlexNet 模型上比 DeepCompression 多节约 1.7 倍的能耗. MeDNN [63] 指出随机的网络剪枝会使模型推理时内存访问开销较大, 削弱模型压缩带来的减小计算量优势, 甚至降低模型压缩后的处理速度. 因此, 他们利用按行或按列组 Lasso (group Lasso) 对神经网络损失函数进行正则化, 使得剪枝后的参数矩阵每行或列具有相同的参数个数. 该方法能够更高效地存储模型并加速稀疏矩阵乘法运算, 在 Nexus 5 手机上可以平均提升 26.5% 的 DNN 推理速度, 和减小 65.9% 的参数存储开销. 不同于上述方法利用人工特征进行剪枝, AMC [64] 通过强化学习策略, 逐层控制每一层的压缩率, 在每一层根据参数绝对值大小进行剪枝, 并采用计算资源限制和分析准确度限制分别作为强化学习的奖励函数. AMC 在 DeepCompression 压缩的基础上再减少了 75% 的浮点运算数, 并且在 ImageNet 数据集 [162] 上得到了高 2.7% 的准确度.

矩阵分解. 神经网络中的大部分计算, 包括卷积计算, 都是通过矩阵乘法进行的. 矩阵分解将参数矩阵和卷积核按照奇异值分解 (singular value decomposition, SVD)、QR 分解等方式进行分解, 然后将其替换为低秩近似. 例如, 将 $M \times M$ 大小的参数矩阵 W 变成 SVD 近似 $W_{M\times M} = U_{M\times k}V_{k\times M}$, 就可以将参数大小从 M^2 减小到 $2Mk$, 将乘法计算量从 M^3 减小到 $2M^2k$. 终端设备上通常不具有训练神经网络的计算资源和数据, 因此相比于需要通过重新训练进行微调的模型剪枝, 基于矩阵分解的压缩方法更常被利用在模型实时压缩的场景中.

DeepX [65] 提出了运行时压缩技术, 在模型推理的过程中把全连接层的参数矩阵进行 SVD 低秩替换. 为了避免较大的准确度损失, DeepX 利用原始参数矩阵和低秩分解的 L2 重建误差来估计这一层神经网络的压缩程度, 并动态限制整个模型的最大误差和. 此外, DeepX 还将一个深度神经网络在本地进行分解, 考虑将各个部分放到设备的 CPU/GPU/DSP/LPC 等硬件模块甚至云端运行. 在进行模型分解时, DeepX 利用贪心算法对模型的分解方案和压缩程度进行搜索, 从而在能耗、运行时间和模型准确度之间进行权衡. DeepX 能在 Tegra K1 开发板上每 500 ms 完成一次大型模型的推理, 并且相比于直接运行节约了 7.12∼26.7 倍的能量. DeepEye [66] 直接将 DeepX 开源的分层压缩工具应用到可穿戴认知辅助任务, 并结合对卷积层和全连接层的分别调度, 加速模型运行和减小能耗. DeepX 的后续工作 SparseSep [67] 基于稀疏编码 (sparse coding) 技术, 通过无监督字典学习的方式进一步压缩参数矩阵, 并且利用稀疏矩阵乘法库对全连接层进行加速. 在损失 5% 准确率的前提下, SparseSep 相比于 DeepX 能够多节约 50% 的内存占用, 在运行速度和能耗上也有一定进步. 不同于 DeepX 等工作针对全连接网络, DeepMon [68] 针对卷积层进行压缩和加速, 利用 Tucker-2 算法直接把卷积层的四维张量分解成 3 个小张量, 并且采用了半精度浮点数, 从而减少乘法运算量.

模型重构. 还有的方法直接修改神经网络的模型结构为移动设备定制轻量级模型, 减小原有架构的参数量和计算复杂度. 例如, MobileNet [69] 将卷积操作用深度可分离卷积替代, ShuffleNet [70] 采用了逐点组卷积 (pointwise group convolution) 和通道随机混合方法, 也有的系统考虑减少卷积的卷积核个数和步长等更简单的架构调整方法 [71], MnasNet [72] 甚至直接根据移动设备上的运行速度指导自

动网络架构搜索. 这些轻量级模型设计虽然可以方便地运行在智能终端设备上, 但是相对于原有架构通常有较大的分析准确度损失, 通常被应用到视频分析流程的初始步骤中.

4.1.2 模型选择

模型选择技术从一系列训练好的模型中, 在运行时动态选择出最合适的模型在本地进行分析, 从而权衡运行速度、准确度和能耗. 由于不同图像的分析难度不同, 能够最快分析出准确结果的视觉模型也不相同. 例如, 在光照较好、背景简单的情况下, 一个快速的小模型就可以识别出视频中的目标; 而在其他情况下则可能必须使用高延迟的复杂模型. 此外, 在多任务并行的情况下, 还需要考虑不同模型对资源竞争带来的一系列问题.

基于输入图像的亮度、对比度等视觉特征, Taylor 等 [73] 通过一系列级联的最近邻搜索算法, 搜索出满足准确度要求的最优分类模型. 实验表明 Taylor 等的方法能在 95.6% 的情况下找到对应输入的最佳模型, 并且相比于单个模型的系统提升了 7.52% 的准确度和降低了 44.4% 的分析延迟. ApproxDet [74] 还同时考虑了当前设备的资源占用情况和输入特征进行模型筛选. 他们将视频分析流程中的不同模型和参数组合作为不同的近似分支, 在考虑输入图片特征、当前资源占用的情况下, 利用二次回归和决策树分别预测不同分支的分析准确率和延迟, 并根据预测结果动态选择每个输入图像的近似分支. 在 Jetson TX2 上的实验中, ApproxDet 比 YOLOv3 模型具有 11.2% 的精度提升和 52% 的延迟下降.

此外, 在视频流分析系统中, 模型选择算法还经常与模型压缩技术结合使用. 例如, DeepX [65] 对压缩程度的搜索过程就可以看作先用模型压缩得到不同模型, 然后进行筛选. AdaDeep [75] 在此基础上更进一步, 综合考虑了参数剪枝和压缩、矩阵分解和模型结构替换这 3 个大类中的 10 种压缩算法在不同层的组合. AdaDeep 利用深度强化学习模型 DQN, 根据用户提供的准确度、延迟、内存、能耗要求, 在线选择最佳的压缩模型. 在 12 个终端设备和 5 个数据集上的实验表明, AdaDeep 能提供最多 9.8 倍的延迟减小, 4.3 倍的能耗节约, 和 38 倍的内存减小, 并且只有小于 2.1% 的准确度下降. MCDNN [76] 则针对增强现实的具体场景, 将视频流分析和多应用并行的特性纳入模型选择的考量. 在某一时间段、某一固定场景下的摄像头所拍摄的视频流, 通常只会出现相似的几个类别. 例如, 在办公室场景下, 某几个人出现的概率可能高达 90%, 而几百个人出现的概率则呈现长尾分布. 基于此观察, MCDNN 利用模型压缩方法生成多个与原始模型功能相同但容量更小的普通模型, 以及多个针对一小部分类别的运行速度极快的定制模型, 并通过测试数据估计其准确度、延迟等指标. 然后, MCDNN 用启发式算法进行模型选择, 并将筛选出的定制模型和普通模型并行执行. 由于定制模型运行速度更快, 在大部分情况下可以提前分析出结果, 降低系统延迟.

在终端设备上动态选择不同的模型, 涉及在运行时对深度学习模型进行切换. 如果将所有模型同时保存在内存上, 可能会超出终端设备的能力; 而如果在切换时再对模型进行加载, 则可能造成较长的加载延迟和能量开销. 因此, NestDNN [71] 将一个原有的深度学习模型训练成一个同时包含多个子模型的多容量模型, 模型切换只需要加载部分参数, 既减小了总的内存占用, 又缓解了模型切换带来的开销. 如图 8 所示, NestDNN 先对原始模型迭代进行卷积层过滤器的剪枝和微调, 直到生成一个最小的种子模型; 然后将种子模型参数固定, 迭代增加过滤器的大小和训练新的参数部分, 直到模型恢复原本的容量, 从而生成多容量模型. 最后, 与其他模型选择算法类似, NestDNN 通过测试数据得出每个子模型的准确度和计算延迟, 并通过启发式算法选择满足延迟要求下精度最大的模型. 多个不同任务和模型上的实验表明, NestDNN 相比于多个独立的模型平均降低了 50% 的内存占用, 并且在参数量较大时有效节约了模型切换的能耗开销.

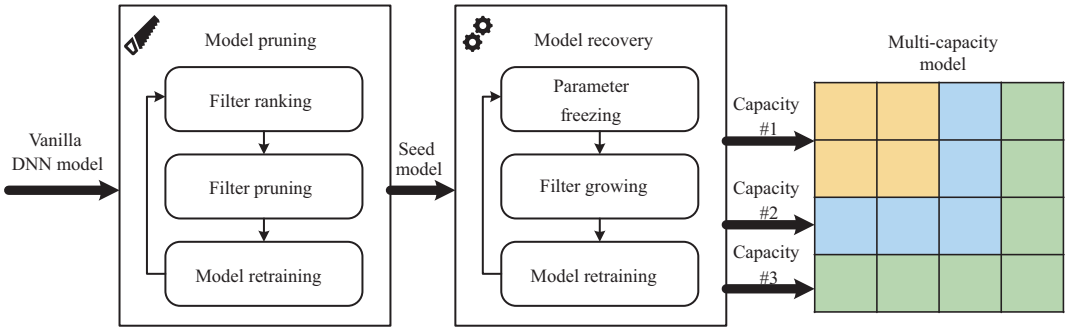

图 8 (网络版彩图) NestDNN 多容量模型训练示意图

Figure 8 (Color online) Multi-capacity model training of NestDNN

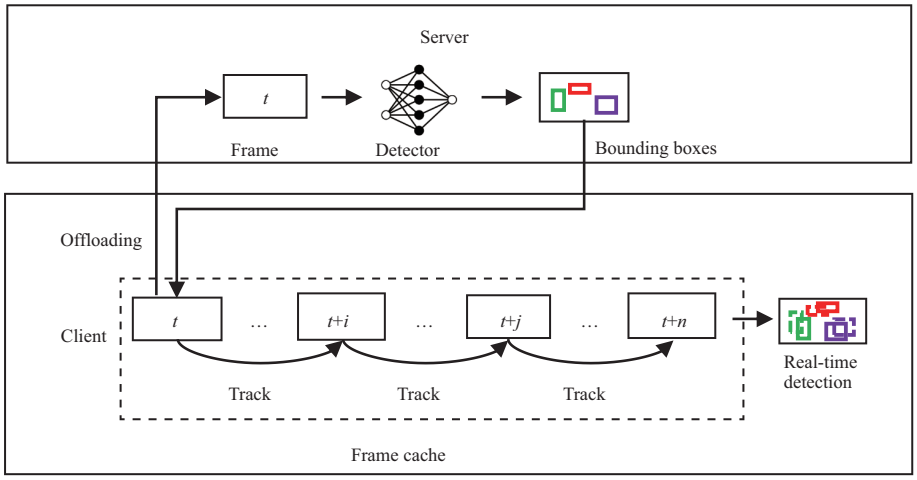

图 9 (网络版彩图) 基于计算卸载的 “检测加追踪” 框架

Figure 9 (Color online) “Detect and track” framework based on computation offloading

4.1.3 本地缓存

实时视频流天然具有时间局部性和空间局部性的特点. 因此, 利用对视频分析结果和中间过程的缓存, 相近视频帧的分析任务通常可以复用之前的分析内容, 通过牺牲部分分析准确度和提高内存开销, 既减小了系统的计算能耗, 又可以提高当前设备的分析帧率.

结果缓存. 许多边缘实时视频流分析系统直接对视频分析的结果进行缓存. 在应用广泛的视频流目标检测任务中, 相邻视频帧之间目标往往只是稍微移动一些位置, 通过轻量级的目标追踪算法就可以准确地从前一帧推测当前帧的目标位置. 因此, 目标检测系统可以每隔 N 帧进行一次目标检测, 在其他视频帧之间利用缓存的检测框结果进行目标追踪, 几乎能将计算开销降低 N 倍. 如图 9 所示, 自 Glimpse [53] 以来, 上述 “检测加追踪” 的思路成为了边缘/云服务器和设备协同进行目标追踪的通用流程: 当前视频帧的实时检测结果由缓存的结果通过目标追踪得到, 而缓存的结果则由边缘/云服务器返回的检测结果进行更新. 只要本地的目标追踪算法可以实时运行, “检测加追踪” 框架就可以实时更新检测结果, 在用户感知中隐藏计算卸载带来的较高延迟. Glimpse 采用了基于光流法的本地追踪, 第一个达到了 30 fps 的手机端检测速度; 之后, Liu 等 [8] 的 EAAR 直接利用了视频编码中的运动

矢量更快速地追踪缓存的结果; EdgeDuet [77] 则采用了 KCF 追踪器 [78], 并且优先更新运动较快的目标. 不同于前述模型采用了基于视频帧的目标追踪方法, Jaguar [79] 利用了谷歌的 ARCore 框架, 基于摄像头位姿估计和惯性测量单元 (inertial measurement unit, IMU) 对目标进行三维世界的追踪. 然后, 他们将三维世界的目标映射到视频帧所在的平面, 并与检测结果进行相互校准, 同时优化了目标检测结果和三维世界追踪精度. 除了目标检测之外, 实例分割算法也可以利用光流估计或者像素块匹配复用前一帧缓存的结果 [39]. 此外, 针对目标识别的 DeepEye [66] 通过径向方差哈希进行视频帧级别的相似度比较, 并复用缓存的识别结果. 结合模型压缩和调度, DeepEye 可以在充满电的小型开发板上连续 33 h 进行每分钟 1 帧的目标检测. EMO [80] 希望利用智能眼镜连续拍摄的单幅人眼图像进行情绪识别. 他们采用了孪生网络思想, 利用情绪识别模块生成的特征图和前一帧特征图进行匹配, 如果一致则复用结果.

除了单个任务场景, 多任务场景下也可以进行结果缓存. Starfish [81] 发现, 大部分多任务实时视频流分析系统, 都共用了相同的底层数据, 而且应用内和应用间都涉及对 OpenCV 等视觉函数库中底层函数的重复调用. 例如, 人脸识别和场景分割都需要把图像下采样并变成灰度图; 照片标签和报警提醒服务都需要人脸检测和识别. 他们在不修改应用和库的源代码的情况下, 把应用对库函数的直接调用变成了对 Core 模块的远程调用. 当收到函数调用后, Core 模块会先检查自己的缓存, 如果发现同样输入的相同函数调用则直接返回结果, 否则再调用相应的视觉库函数. 在谷歌眼镜上的实验表明, Starfish 可以节约 19%~58% 的多任务分析能量开销. 不同于 Starfish 只针对完全相同的图像, Potluck [82] 支持利用不同但相似的图像, 并且将可以复用的函数从普通的图像处理扩展到了深度神经网络. 在同时运行两个增强现实和一个图像识别应用时, Potluck 提升了 2.5~10 倍的处理速度.

中间过程缓存. 还有许多系统对视频流分析过程的中间结果, 例如卷积神经网络的特征层, 进行缓存和复用. CBInfer [83] 针对像素变化很少的固定摄像头上的视频实时实例分割问题, 提出了基于像素变化的卷积算法. 具体来说, CBInfer 对每一个卷积层的输入特征图进行像素比较, 提取出其中数值差异大于阈值的像素位置, 然后仅对这些位置相关的行和列进行卷积所需的矩阵乘法运算, 最后将此运算结果和上一帧的输出结果进行合并, 从而得到本层的新输出结果. 该方法在不需要任何模型训练的情况下, 能够提升视频分割 8.6 倍的运行速度, 节约 10 倍的能耗, 仅降低 0.1% 的像素准确度. DeepMon [68] 则对每个卷积层输入特征图上同一位置的图块, 用颜色分布直方图的卡方距离进行比较, 如果距离低于阈值, 则复用该特征图块对应的输出. 该方法能够在背景像素少量变化的情况下复用部分结果, 解决了 CBInfer 只能用于固定摄像头的问题. DeepCache [84] 在 DeepMon 的基础上进行了两点改进: 第一, 将固定位置块匹配变成了动态块匹配, 增加了缓存命中率, 且不需要修改神经网络架构; 第二, 从比较每个卷积层输入变成了只比较一次原始图像相似度, 减少了比较查询的开销. 如图 10 所示, 由于仅通过原始图像进行匹配, 卷积层的可复用范围会逐层变小, 因此需要匹配尽可能大的连续矩形图像块. 具体来说, DeepCache 运用了视频编码中常见的钻石搜索 (diamond search) 算法进行原始图像的块匹配, 然后以所有图像块的平均移动向量为基准匹配尽可能多的视频块, 从而得到尽可能连续且相似的视频块. 在 UCF101 数据集上, DeepCache 匹配算法的缓存命中率为 69.5%, 相比于 DeepMon 提升了 50%. 经过多个数据集的实验, DeepCache 相比于没有缓存平均减小了 18.2% 的延迟, 而 DeepMon 仅减少了 8.9%.

4.1.4 视频帧过滤

视频帧过滤在逻辑上可以被等效为图 3 所示视频分析流程中的预处理部分, 只有过滤后的视频或视频帧会进行后续的检测、识别等分析. 通过筛除冗余的输入, 该方法可以在损失较少精度的情况下,

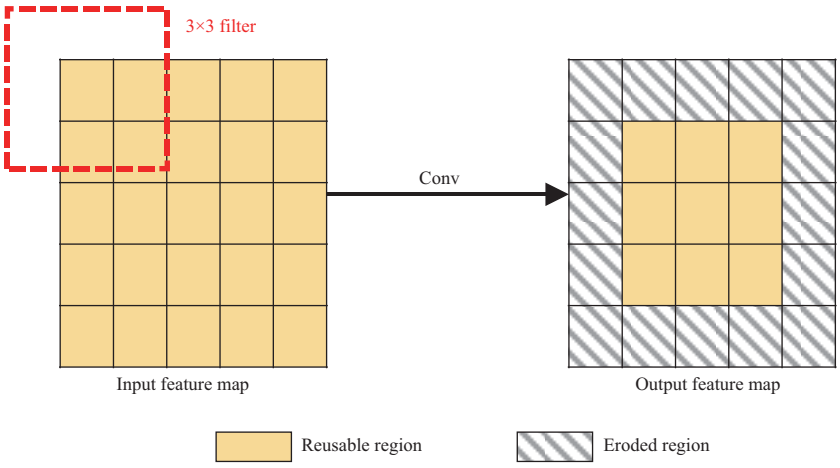

图 10　(网络版彩图) 卷积缓存的可复用部分逐层变小的示例

Figure 10　(Color online) An example of cache erosion at the convolution layer

有效地减少分析的延迟和能耗开销. 视频帧过滤方法所需的计算资源比完整的视频流分析要小一到两个数量级, 且处于整个视频分析流程的上游, 因此通常可以应用在离视频源最近的终端设备上. 此外, 在设备 – 边缘 – 云端协同场景下, 对输入进行过滤还可以减少终端设备对边缘/云服务器的分析请求, 起到节约带宽和扩大服务规模的作用.

基于视频帧变化过滤. 一系列工作通过分析视频中相邻帧的变化情况, 仅对变化较快的视频帧进行分析. Gammeter 等 [85] 基于云端协同进行增强现实的目标识别, 在本地通过视觉方法追踪识别结果, 当视觉追踪和基于 IMU 的追踪出现矛盾时, 才上传新的视频帧到云端进行分析. 同样采取 “检测加追踪” 方案的 Glimpse [53] 则比较视频中相邻两帧的像素差值来估计视频帧的变化情况, 如果发生变化的像素数目超过阈值才上传视频帧进行云端目标识别. Reducto [86] 进一步指出, 不同视频分析任务应该采用不同的图像比较特征进行过滤, 且过滤的阈值应该根据视频内容动态调整. 为了更好地指导摄像头进行分析, Reducto 先让云服务器对 10 min 长度的原始视频流进行分析, 并利用聚类算法得到从视频片段变化情况映射到最佳过滤阈值的哈希表. 基于该哈希表, 摄像头可以通过最近 30 s 的视频变化情况动态调整视频帧过滤的阈值, 如果查询失败, 则上传最新的原始视频流供云端重新训练. 在保持 90% 分析准确率的前提下, Reducto 在多个监控视频数据集上过滤掉了 51%~97% 的视频帧. 不同于上述工作利用底层的图像特征进行比较, DorT [87] 采用了孪生神经网络模型, 利用同样的卷积神经网络提取相邻视频帧的特征图, 通过深度特征比较视频帧变化情况. 具体来说, DorT 将孪生神经网络的特征图作为输入, 训练了一个简单判别器, 用以判断当前帧需要追踪还是检测. 不管判别结果为追踪还是检测, 都可以继续复用孪生神经网络提取出的特征图.

基于本地分析过滤. 另外一系列工作利用定制化的小模型对单个视频帧的内容进行独立分析, 从而决定是否过滤. FilterForward [88] 训练了多个二元分类器, 每一个分类器都利用一个共同 MobileNet 模型的中间层预测每一类别是否存在. 当连续 5 帧出现两次阳性判断之后, FilterForward 才会把对应的视频片段传给云端进行分析. FFS-VA [89] 针对大规模视频流分析, 采用了三级过滤的思路, 在三级过滤完之后才会用标准模型进行分析. 第 1 级将当前帧与之前数帧的平均值比较均方误差, 过滤掉只包含背景的帧; 第 2 级用一个三层的小型卷积神经网络判断当前帧是否存在目标, 过滤掉没有目标的帧; 第 3 级则利用只有 9 层卷积的小型 YOLO 变体, 过滤掉目标数量小于阈值的帧. 这三级过滤器相

比于标准的 YOLOv2 模型分别提升了 300, 30 和 7 倍的处理速度, 三者一同提升了整个系统 7 倍的吞吐量, 仅损失了 2% 的分析准确度. Frugal following [90] 针对增强现实中的目标追踪, 采用了两层过滤: 第 1 层利用归一化交叉相关系数判断光流追踪器是否出现错误, 第 2 层则通过随机森林模型预测是否出现新的目标, 出现任意一种情况就启动目标检测模型. 他们的过滤方法能够在损失 7.3% 准确度的情况下, 相比基于 tiny YOLO 的检测加追踪方式节约 82% 的能耗. Wang 等 [91] 的工作旨在通过帧过滤减少无人机的带宽开销, 提出了 4 种可以相互组合的策略: 提前退出、即时学习、回退和情景感知策略. 其中, 提前退出策略利用基于 MobileNet 的模型判断视频帧是否重要; 即时学习和回退策略分别试图减少提前退出模型的误报和回溯过滤掉的结果; 情境感知策略基于分析任务动态选择合适的简单过滤器. 此外, Wang 等 [59] 结合可穿戴认知辅助的应用场景, 利用检测的目标任务的面部运动状况和 IMU 数据, 训练支持向量机模型判断用户是否正在进行互动, 仅在用户处于主动互动状态时上传视频帧进行分析.

值得注意的是, 虽然基于模型的办法可以相对更准确地对视频帧进行分析和筛选, 但是大部分商用摄像头上的计算资源极其有限, 即使是压缩并且定制化的小模型都难以实时运行. 因此, Glimpse 和 Reducto 等基于底层视觉特征进行视频帧过滤的方法可以更容易地进行部署和推广到网络的最边缘端. 而 FilterForward 等基于模型的方法则更适合具有一定计算能力的智能边缘计算设备.

多摄像头过滤. 除了针对单个视频流, 还有的工作对多摄像头场景进行协同过滤. Vigil [44] 考虑了基于无线网络的云边结合的分布式监控视频分析系统, 每个边缘计算节点服务多路相邻视频流, 边缘计算节点在进行视频分析的同时, 需要把分析结果和与之相关的视频帧传给云端. 为了减少边缘计算节点占用的上传带宽, Vigil 同时采用了两种过滤方式: 基础方式是先找到滑动窗口内平均检测到目标最多的视频流, 然后上传其中目标个数有变化的帧. 进阶方式则利用已知的相机位姿和视角重叠关系, 判断相邻视频流是否重复检测同一目标, 然后利用该信息将基础方式错过的目标所在视频帧进行上传. 针对人脸识别和行人重识别任务, 在几乎不损失精度的情况下, Vigil 提出的视频帧上传协议相比于直播整个视频流节约了 5~200 倍的带宽. Spatula [92] 针对多摄像头网络上的实时目标追踪任务, 先离线学习摄像头间的时空依赖关系, 即一个目标离开任意摄像头范围后到其他摄像头的概率和时间分布; 然后在进行跨摄像头追踪时, 根据时空依赖关系过滤掉不太可能出现目标的视频帧, 从而减少进行目标重识别的次数. 相比于逐帧进行跨摄像头目标追踪, Spatula 在上百路视频的仿真数据集上能节约 23~86 倍的计算开销.

Panoptes [16] 和 Jang 等 [93] 则提出了摄像头虚拟化的概念, 把一个摄像头虚拟化成多个摄像头, 通过筛选和整合视频帧为不同应用同时提供服务. 其中 Panoptes 根据多个应用的需求和现场情境控制摄像头转速和方向, 尽量满足最多的应用, 同时维护能覆盖全景虚拟视角的视频帧, 根据应用的视角要求发送对应视角的最新历史帧. Jang 等则根据不同应用的需求和场景中的语义信息, 对发送给不同应用的视频流采取不同的操作. 例如, 行人追踪应用需要更高的帧率; 而车牌识别应用则需要进行边缘增强, 并在光线较暗情况下进行亮度增强. 每个应用的需求被人工记录到一个知识库中, 从而自动指导摄像头的筛选和处理逻辑.

4.1.5 小结

终端设备层次的技术主要是为了解决设备计算资源不足和视频分析模型资源开销较大之间的矛盾. 其中模型压缩和模型选择通过减少单次分析的计算复杂度和内存开销来缓解这一矛盾; 本地缓存和视频帧过滤则从减少本地的完整分析次数来缓解这一矛盾. 显然, 每种技术对终端设备资源的优化维度各有不同, 且在减少资源开销的同时也都在模型分析准确度上付出了一定损失. 因此, 一个能够实

表 2 终端设备层次技术相关文献总结对比

Table 2 Summary and comparison of selected studies of device level technologies

Ref.	Application	Key method(s)	Pros.	Cons.
[63]	Classification	Structured sparsity pruning	• Reduce model size • Accelerate memory access	• Handcrafted Lasso regularizer
[65]	General	SVD layer compression; DNN architecture decomposition	• Save energy cost	• Only support CNN • Poor resource estimator
[66]	WCA	DNN layer execution scheduler; local caching scheme	• Save energy cost • Accelerate DNN inference	• Low inference frequency
[73]	Classification	Cascaded KNN-based model selection	• Low-level vision features • Accelerate DNN inference	• Ignore model switching overhead
[74]	Detection	Multi-branch DNN kernel scheduling	• Consider available resource • Improve accuracy & speed	• Bad GPU utilization
[75]	General	DRL-based model compression	• Tradeoff on latency, energy, memory, and accuracy	–
[76]	AR	Cross DNN feature map sharing; approximate model scheduling	• Novel multi-program streaming DNN scenario	• Only one time profiling
[71]	General	Multi-capacity DNN model	• Tradeoff on resource & accuracy • Low model switching overhead	• Computation overhead for model generation
[53]	Continuous recognition	"Detect & track" paradigm	• Reduce computation overhead • Hide response delay	• Not fast enough for AR
[8]	AR	Motion vector-based tracking	• 60 fps fast local tracking	• More accuracy loss
[82]	AR	Reuse DNN across similar images	• Save energy & computation	–
[68]	Classification	Feature map caching between frames; convolution decomposition	• Allow large DNN execution on mobile • Block granularity feature map reuse	• Run DNN at 1∼2 fps only • Reuse blocks at same position
[84]	General	Dynamic block matching & feature map reuse between frames	• Higher cache hit rate for feature map reuse	–
[86]	Surveillance	Server driven on camera frame filtering	• Filter $>$ 50% frames with little accuracy loss	• Only use low-level vision features
[90]	AR	Pixel-based & random forest-based frame filtering	• Save energy & bandwidth	–
[59]	WCA	Offloading decision based on human-computer interaction	• Application-aware workload adaptation	• Cannot resist malicious users
[44]	Surveillance	Frame & feed selection with overlapping cameras	• Save bandwidth • Solid system implementation	• Only support traditional CV algorithms
[92]	Surveillance	Cross camera spatio-temporal frame filtering	• Reduce computation overhead • Capture camera correlation automatically	• Ignore network bandwidth • Assume non-overlapping cameras
[16]	Surveillance	Steerable camera view virtualization	• Multi-tenancy support	• Cannot support real-time applications

际运行的系统必然需要在各种关键指标优化和技术选取上进行权衡, 从不同维度提升系统表现. 我们将涉及终端设备层次技术的部分代表性工作的应用场景、核心方法和优缺点总结在了表 2 中. 可以看到, 模型压缩和选择技术经常同时出现, 同样可以用在视频分析之外的其他通用任务中. 而视频帧过滤和本地缓存则基于视频流的冗余特性进行了针对性优化.

具体来说, 模型压缩技术利用了深度学习模型的冗余性, 通过模型剪枝、分解等方法, 降低模型参数数量和复杂度并保持其分析功能. 其主要目的是降低模型对内存资源的占用, 使普通摄像头或型号较老的智能手机也能够用有限的内存加载运行深度学习模型. 然而, 模型压缩通常会修改模型结构, 降低其并行性和内存访问效率, 对模型运行速度提升并不明显. 即使通过结构化剪枝、稀疏矩阵乘法库等方法进行加速, 仅依靠模型压缩也难以在普通终端设备中进行实时视频分析. 模型选择技术与模型压缩技术相辅相成, 后者可以提供不同大小的模型, 前者则根据设备资源和模型性能预测实时挑选合适的模型进行分析. 其优势在于可以在尽量不损失精度的情况下优化设备能耗和分析延迟; 劣势在于需要在资源有限的终端设备上储存多个模型并进行切换, 且多模型训练的计算开销较大. 本地缓存技术利用了视频分析任务的时间局部性特点, 直接复用视频分析结果或部分中间结果, 从而提升同一设备的视频分析帧率. 该方法尤其适用于增强现实场景, 既能在单个应用中提升分析帧率适应人眼感知, 又能在多任务场景下减少系统整体开销. 其劣势在于缓存分析结果, 特别是神经网络的中间运行结果会带来较大的内存占用, 对设备性能有一定要求. 视频帧过滤技术利用了视频流内容的时空冗余特性, 筛选掉信息含量不大的视频帧. 有效的过滤能够将视频分析速率提升一到两个数量级, 且对计算复杂度要求不高, 可以用于视频分析流程最上游的预处理阶段, 适用于各种应用场景.

除了上述算法和系统上的改进之外, 也有的工作考虑在智能终端设备上部署更强大的专用硬件, 例如 FPGA, ASIC, NPU 等. 然而, 由于成本和部署难度, 这些专用硬件目前很难推广到真实世界的视频分析应用中, 因此本文不作进一步分析. 感兴趣的读者可以参考 Deng 等 [163] 的综述.

4.2 协作层次

协作层次包括了任务卸载、网络协议和隐私保护等技术.

4.2.1 任务卸载

将视频分析流程按照图 3 中的各个任务进行切分和卸载, 是基于边缘计算的实时视频分析中常见的计算范式. 例如, Glimpse [53] 等早期方法采用的 “检测加追踪” 模型, 或者传统的云端视频分析, 就是典型的固定规则任务卸载. 然而, 固定的任务卸载方式并不一定能取得最好的视频流分析效果. 一方面, 将复杂的计算任务交给计算能力更强的上层设备执行, 既可以减少计算的时间开销, 又可以提高分析的准确度; 另一方面, 在上传任务和返回结果过程中的通信延迟, 又有可能造成最终的端到端时延提升, 进而间接影响分析准确度. 因此, 系统通常需要对延迟、带宽、分析精度, 以及能量开销作出权衡, 并根据可能的网络带宽波动, 综合考量将哪些计算任务分配到云边端的哪一层和哪个设备上. 此外, 由于深度学习模型往往在视频流分析任务中占据了最多的计算资源, 许多工作在进行逻辑模块划分的基础上, 进一步考虑了对深度神经网络进行模型粒度的分割和卸载. 我们对其中有代表性的方法, 按照纵向卸载和横向卸载分别进行讨论.

纵向卸载. 许多工作考虑将任务纵向卸载到计算能力逐级增强的终端设备、边缘节点和云端, 在满足各种资源约束的前提下, 权衡视频分析准确度和端到端延迟.

其中大部分工作仅考虑存在一个边缘/云服务器的情况. 如图 11 所示, 典型的 DeepDecision [52] 根据网络状况动态决定在本地运行压缩的目标检测模型还是卸载到云端. DeepDecision 首先通过离线

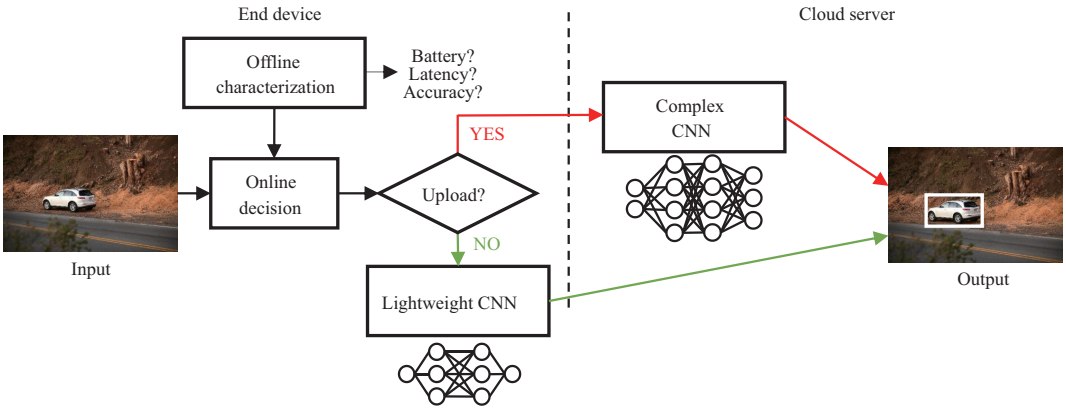

图 11　(网络版彩图) DeepDecision 的任务卸载框架

Figure 11　(Color online) Task offloading framework of DeepDecision

实验测量出视频分析的传输延迟、模型精度等指标, 与视频分辨率、压缩比特率、模型大小、是否卸载等变量的关系; 然后在运行时周期性地根据当前网络状况用贪心算法求解优化问题, 调整下一个时间窗口各个变量的选择, 最大化分析的视频帧数和准确度. 在三星 S7 手机的测试中, DeepDecision 能够快速适应网络带宽和延迟的变化, 并且始终保持高于基准算法的准确度. VideoEdge [17] 针对大规模监控场景, 假设系统需要持续服务多路视频, 设备和边缘服务器上计算资源有限, 边缘和云端的带宽有限, 因此需要将每个视频流的分析任务在设备、边缘服务器、云端进行卸载. VideoEdge 将 CPU 核心数和网络带宽作为待分配的资源, 分别测量出不同模型和不同任务卸载方式下的视频分析准确度与资源占用, 然后将其放进一个二元整数规划框架中最大化总分析准确度. 通过一系列启发式搜索算法, VideoEdge 相比于平均分配资源提升了 25.4 倍总准确度, 并且在最优解准确度的 93%~96% 范围内, 但是忽略了视频分析的延迟. 类似地, Ding 等 [94] 将频谱分配也纳入资源考量, 通过随机优化算法对任务卸载方式进行求解; Wang 和 Xie [95] 考虑通过控制终端 CPU 的频率权衡分析速度和能耗. 不同于上述系统直接利用测量结果预测分析效果, JCAB [96] 给出了准确度与分辨率、帧率关系的非多项式分析模型, 并利用马尔可夫 (Markov) 近似和卡鲁什 – 库恩 – 塔克条件 (K.K.T condition), 把针对长时间平均准确度和能量开销的复杂问题, 拆解成一系列小的近似问题实时求解. 他们在数学上证明了自己的方法能够以亚线性速度求解, 通过仿真进行了验证.

还有的系统考虑到了同时存在多个边缘服务器的情况, 针对不同服务器的资源状况进行选择性卸载. Liu 等 [97] 在终端设备和边缘服务器之间加入了一个被称为边缘配置器的专门设备, 配置器负责给多个终端设备分别分配服务器和确定视频上传帧率. 他们对视频分析流程中的网络延迟、处理延迟和分析准确度等值与视频帧率的关系进行建模, 然后通过混合整数非线性规划问题最大化各个任务的总分析准确度和最小化总延迟. 仿真实验表明, 在相同分析准确度下, 该方法相比于固定帧率且随机选择边缘服务器的方法减少了 26% 的端到端延迟. Trinh 等 [98] 还利用机器学习算法对卫星图像进行分析, 估计设备间的物理位置距离, 从而辅助系统在多个边缘服务器和云端之间作出卸载决定, 减小系统总能耗. 由于许多情境下需要对高清摄像头的视频进行复杂的高清图像分割、多人姿态估计等任务, 而一个边缘计算节点的计算能力可能不足以高效完成任务, Elf [99] 将一个视频帧分块并卸载到不同的服务器进行并行分析. 为了更好地进行图像分块, Elf 设计了轻量级的基于注意力机制和长短时记忆神经网络 (long short-term memory, LSTM) 的模型, 基于历史数据帧的候选区域 (region proposal) 预测当前帧的目标区域范围. 然后, Elf 对各个边缘服务器的计算资源以及每个候选区域的计算开销进行估

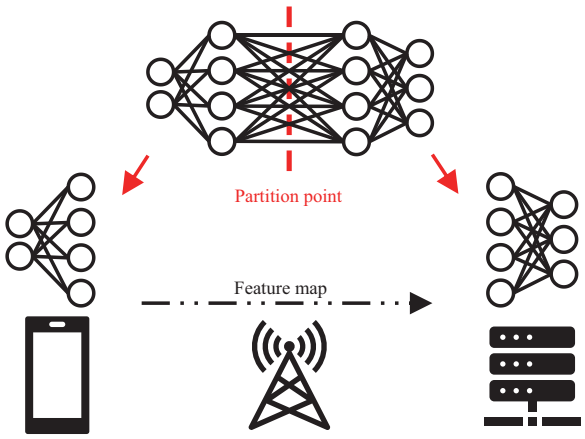

图 12 (网络版彩图) 纵向模型分割示意图

Figure 12 (Color online) Vertical model partition

计, 并通过启发式算法将图像进行分块然后分别发送到各个对应的边缘服务器, 达到负载均衡的效果. 他们用多个智能手机和开发板作为终端设备, 5 个包含 GPU 的电脑作为边缘服务器, 并在仿真的网络环境下进行了实验测试. 结果表明, Elf 相比于将任务上传给单个服务器提升了 5.43 倍处理速度, 并减小了 53.6% 的带宽占用, 而考虑了本地过滤方法的 FilterForward [88] 仅能提升 1.56 倍处理速度.

纵向分割是指我们可以将深度神经网络看作一个计算图, 在这个计算图中找到一个合适的分割点 (partition point) 将网络划分成多个顺序执行的部分. 如图 12 所示, 纵向分割可以将一个神经网络的部分计算任务从终端设备卸载到边缘/云服务器上分别执行, 并通过网络传输中间特征层. 前述纵向卸载方法仅将深度学习模型作为黑盒整体考虑, 而对模型进行纵向分割之后可以更细粒度地卸载计算任务, 充分利用设备计算能力和网络资源. Neurosurgeon [100] 发现, 在 3G 网络下将图像传输给云端进行分析所需的传输延迟和能耗甚至大于在手机上执行所有分析, 而 AlexNet 等深度学习模型中部分全连接层的输出比原始图片体积小很多. 他们将神经网络中常见的卷积层、全连接层、池化层等分别放到终端和服务器上进行测试, 并利用回归模型拟合出每层参数和能耗、延迟的关系. 然后, 他们将网络状况纳入考量, 以层为粒度对神经网络进行分割, 从而最小化整个分析过程的端到端延迟和手机上的能量开销. 多种网络条件下的测试中, Neurosurgeon 相比于纯服务器分析方案平均减少了 67.7% 的延迟, 以及 59.5% 的能耗. Li 等 [101] 的方法与 Neurosurgeon 类似, 但是考虑了多个边缘服务器同时服务多个摄像头的情况, 并将多摄像头模型分割问题转化为非线性规划, 在确保每个摄像头满足分析延迟限制的条件下支持尽可能多的视频路数. Teerapittayanon 等 [102] 提出了 BranchyNet, 如图 13 所示, 研究了深度学习模型提前退出机制. 如果一个退出点能够高概率地完成分析任务, 则后面的网络不需要继续运行. 通过将模型每个退出点的输出相加进行联合训练, BranchNet 的每个退出点都能够具有一定的分析能力, 而且层数越深的退出点分析准确度越高. 在 BranchyNet 的基础上, Teerapittayanon 等 [103] 提出了 DDNN, 将提前训练好的 BranchyNet 网络按照退出点进行分割, 分别放在多个终端设备和多层边缘/云服务器上. 由于提前退出机制, DDNN 可以起到对输入进行过滤的作用, 在损失有限准确度的情况下减少系统的分析延迟和带宽占用. 在多个设备共同进行分析的实验中, DDNN 甚至能节省 20 倍的带宽占用. Edgent [104] 将 Neurosurgeon 和 DDNN 进行了结合, 同时考虑了模型分割点和提前退出点的选择. Edgent 首先训练很多个有不同退出点的模型, 然后按照遍历方式, 找出满足时延条件下平均准确度最高的退出点和分割方式. 实验表明 Edgent 能够很好地根据不同网络带宽和延

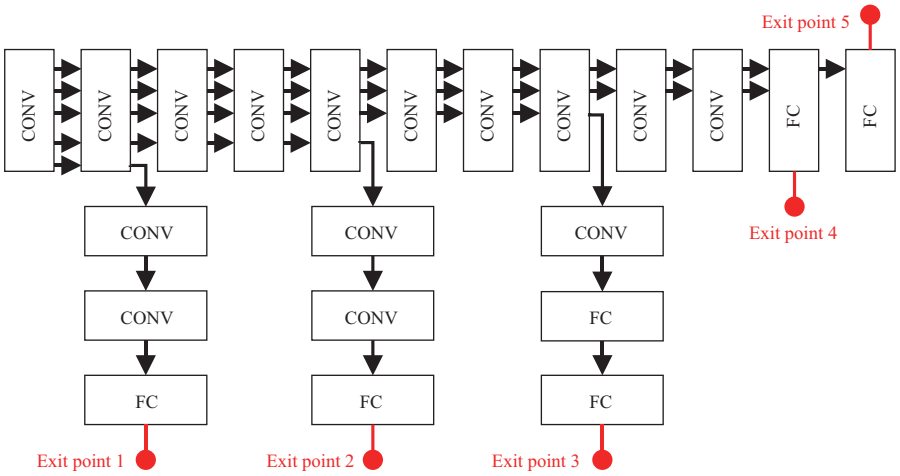

图 13　(网络版彩图) 模型提前退出机制示意图 (包含 5 个退出点)

Figure 13　(Color online) Model early-exit mechanism (with 5 exit points)

迟要求选择对应的模型和切割方式. SPINN [105] 采用了和 Edgent 类似的模型分割和提前退出协同策略. 考虑到输入数据的复杂性, SPINN 使用了渐进式推理网络 (progressive inference network), 并在运行时根据系统资源和网络状况动态决定提前退出置信度和分割位置. 而且, 他们在设备端固定了一个提前退出点, 即使网络断开也可以确保模型正常运行. SPINN 相比于 Edgent 提升了 1.15∼3.09 倍吞吐量, 同时平均减少了 68.64% 的服务器计算时间. Ren 等 [106] 则强调了在波动的网络环境中动态决定模型分割方式. 他们周期性地采集当前网络状况和设备计算能力, 利用基于策略学习的深度强化学习算法对预训练的 BranchyNet 模型进行分割, 最小化延迟和带宽占用. Ren 等在 5G 测试网络上进行了实验, 表明其算法在能耗、延迟和吞吐量上都优于固定分割方式. 此外, IONN [107] 指出, 在游牧服务模式的边缘计算过程中, 模型不一定提前在服务器加载好, 因此系统还需要考虑上传深度学习模型带来的带宽和延迟. IONN 先按照计算资源和网络情况决定模型分割计划, 然后将需要上传的模型分成多个部分增量上传到边缘服务器, 服务器在收到部分模型之后即可开始完成部分模型推理任务.

对于按照顺序执行每一层的链式神经网络结构, 可以通过线性遍历方式快速找到切割点. 然而, 随着深度学习技术的发展, 许多应用广泛的模型, 例如 ResNet [164], Inception [165] 等, 都呈现出如图 14 所示更复杂的模型结构. DADs [109] 将神经网络视作一个有向无环图, 将模型在边缘和服务器端分割视作找到图的一个割集, 并根据网络和计算资源状况提出了两种分割算法. 当网络状况良好、计算资源丰富时, DADs 直接优化每个视频帧的处理延迟, 将模型分割转化为最大流最小割问题求出最优解. 当网络发生拥堵时, DADs 尝试在预估的网络带宽条件下通过模型分割最大化分析吞吐量, 并通过近似搜索算法进行求解. 实验表明, DADs 在复杂的神经网络结构下, 能比 Neurosurgeon 提升 66%∼86% 的处理速度, 以及 76%∼87% 的吞吐量. 同样面对复杂网络结构的问题, DeepWear [110] 则通过剔除轻量级计算和寻找网络中重复的子结构等方法, 将有向无环图进行化简, 然后利用启发式搜索的方法找到最佳模型分割位置. 该算法能够自动将具有 1096 层的 GoogLeNet [166] 简化为只有 35 个可分割节点的链式结构. Wang 等 [111] 同时考虑了模型选择、模型压缩, 和模型在设备、边缘、云端的分割. 他们利用增强学习方法, 根据网络状况和模型的延迟、准确度特性, 考虑原始模型每一层进行压缩或模型分割的可能性, 得到一个可以快速切分的模型树. 模型树的每一个分支代表一种模型压缩和分割方式. 需要在线运行神经网络时, Wang 等再根据当前网络状况和计算资源, 通过基于 LSTM 的控制器

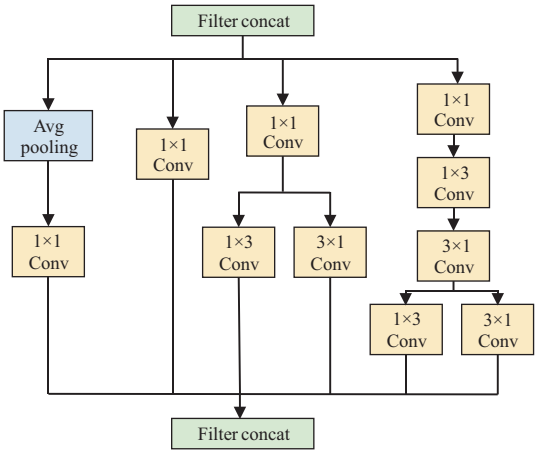

图 14　(网络版彩图) Inception v4 部分网络架构

Figure 14　(Color online) Part of Inception v4 network architecture

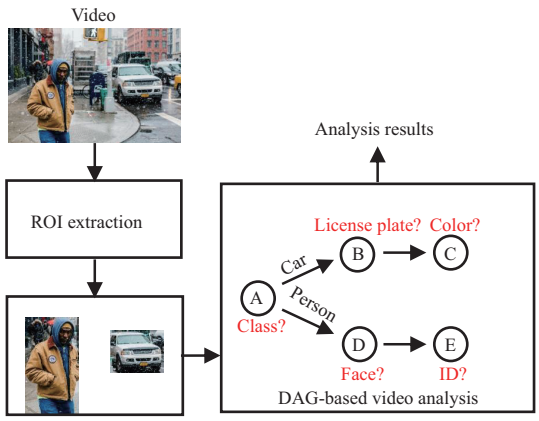

图 15　(网络版彩图) 基于有向无环图的复杂视频分析流程

Figure 15　(Color online) DAG-based complex video analytics pipeline

快速选择模型树分支. 相比于 DADs, Wang 等在以 VGGNet 为原始模型的实验中减少了 30%~50% 的延迟, 仅损失了 1% 的分析准确度.

横向卸载.　除了纵向任务卸载, 许多工作同样考虑了同一层级中不同设备之间的横向任务卸载, 试图更高效地利用分布式边缘计算系统中的闲置资源.

其中, LAVEA [112] 基于经典的设备 – 边缘 – 云端架构, 在边缘计算节点之间进行了任务卸载. 首先, 系统通过测量估计任务在设备和边缘的运算时间, 以最小化端到端延迟为目标, 将部分任务从设备卸载到边缘节点. 如果某个边缘计算节点任务过载, 在考虑向云端卸载之前, 会尝试将过载的任务卸载到附近的空闲节点, 避免云端通信带来的延迟开销. 边缘节点之间通过一系列通信协议实时记录并预测周围节点的响应时间, 优先将任务卸载给预计响应时间最小的节点. 在合成的视频分析数据集上的实验表明, LAVEA 提出的节点间卸载方法相比最短传输时间优先、最短等待队列优先等任务分配算法, 具有更好的平均端到端延迟和吞吐量. 与 LAVEA 类似, SurveilEdge [113] 根据每个节点的预测延迟和队列长度优先将任务卸载给总延迟最小的节点, 而且利用了对数正态分布对每个节点的计算延迟进行实时拟合和估计. 此外, SurveilEdge 还离线将每个摄像头的类别分布作为特征进行聚类, 为不同类别的摄像头微调模型参数, 只需要一分钟就可以使模型达到 92% 的准确度. 根据作者采集的 170 h 监控数据, 其提出的横向卸载相比于边缘节点独立分析减少了 93.6% 的延迟, 相比于固定的云边卸载方案减少了 85.7% 带宽占用. Long 等 [114] 直接将资源富余的智能手机当作边缘计算节点, 辅助摄像头完成延迟敏感的视频分析任务. 他们利用优化算法分别求解一个分组问题和匹配问题. 系统将终端设备进行分组, 摄像头则周期性地将压缩后的视频片段发送给匹配的分组进行处理, 相同分组内的设备分别并行负责处理视频片段中的一部分, 并将最终的分析结果发给云端. 仿真实验中, Long 等提出的分组横向卸载方法比不分组和随机分组的卸载方法分别提高了 19% 和 3% 的准确度. Jiang 等 [115] 提出可以将作为视频源的智能摄像头考虑为多设备的计算集群, 并指出了摄像头间横向卸载带来的资源利用率提高、总吞吐量增大、分析精度提高、开发接口统一等益处. Distream [116] 针对如图 15 所示的基于有向无环图的复杂视频分析任务, 同时考虑了摄像头间负载均衡和摄像头与边缘服务器的任务划分, 在满足应用对延迟要求的前提下尽可能最大化整个系统的吞吐量. Distream 采取了先在摄像头端

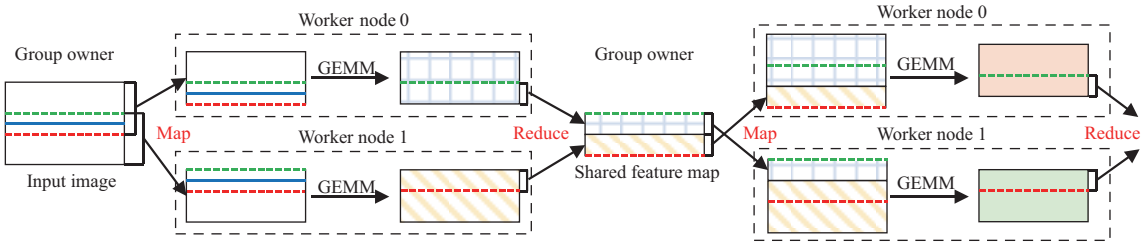

图 16 (网络版彩图) MeDNN/MoDNN 基于MapReduce 的分布式 DNN 运行流程

Figure 16 (Color online) MapReduce-based distributed DNN inference pipeline for MeDNN/MoDNN

负载均衡, 再在摄像头和边缘服务器之间进行任务划分的方式. 具体来讲, Distream 训练了一个 LSTM 模型用于预测每个摄像头接下来需要分析的 ROI 个数, 然后在不同摄像头间进行任务卸载, 试图使每个摄像头的计算资源占用率接近. 在 6 个智能摄像头和一个边缘服务器上的实验表明, Distream 提升了 1.6~2.9 倍的总吞吐量, 并且减少了 99.2%~99.4% 的平均延迟. 此外, Distream 没有考虑通过调整参数对准确度、吞吐量进行权衡, 其提出的摄像头负载均衡模型可以直接运用到 VideoEdge [17] 等动态调整参数的框架中, 进一步提高资源利用率.

还有部分工作对深度神经网络模型进行了横向分割, 从而在不同设备间进行更细粒度的计算任务卸载. 不同于纵向分割将深度学习模型切分为顺序执行的子部分, 横向分割将模型切分为数个可以并行执行的部分, 然后将其卸载到同一层级的计算设备中. 该方法主要利用了神经网络中计算量最大的卷积操作所具有的局部性特点, 即下一层特征图中某一个区域, 只和输入层中的部分区域相关. 如图 16 所示, MoDNN [117] 借鉴了MapReduce [167] 的思想, 源节点将卷积层的输入进行分割, 然后分别卸载到不同计算节点上并行执行, 每个节点执行完卷积运算之后将结果发回给源节点, 源节点合并完本层所有结果之后再次对下一层卷积进行分割和卸载. 对于卷积层, MoDNN 按照输入矩阵的较长边进行平均分割, 均衡地卸载其计算任务; 对于全连接层, MoDNN 按照 SVD 和按行聚类算法进行分解, 减少参数传输的通信开销. 在 2~4 个 Android 手机的实验上, MoDNN 对神经网络的计算起到了 2.17~4.28 倍的加速. 作为 MoDNN 的改进版本, MeDNN [63] 提出了贪婪二维分割算法, 根据每个设备的计算能力进行静态负载均衡. 他们通过遍历方式离线选择输入矩阵的最佳划分, 并结合了第 4.1.1 小节中提到的结构化剪枝降低参数量和加速计算. DeepThings [118] 认为 MoDNN/MeDNN 的逐层分发然后合并的方法运行效率较低, 在每一层等待所有设备的输出结果会减少并行效率和增加通信开销. 因此, DeepThings 提出合并分割方法, 将神经网络中的最底层的所有卷积进行合并分割. 具体来说，其首先将最终输出的特征图分割为多块, 然后把每块特征图对应的原始输入图像块分别卸载到不同设备上执行卷积推理, 并且动态均衡设备间的负载. 由于一个输出区域对应的输入区域随着层数加深而逐渐增大, 每个设备分割到的输入区域会有部分重叠, DeepThings 通过任务调度算法, 在较小的通信开销下使设备间共享部分计算结果. DeepThings 相比于 MoDNN 提升了 80% 的吞吐量, 减小了 41% 的平均延迟. Zhou 等 [119] 将 DeepThings 的合并分割方法进行扩展, 动态选择将哪些连续的卷积层合并分割. 他们综合考虑通信和计算开销, 通过基于动态规划的搜索算法找到最佳的合并层数和分割位置, 减少模型的端到端延迟, 最终在 8 个开发板组成的实验环境下取得了 3.7 倍的加速.

4.2.2 网络协议

任务卸载主要考虑如何分割计算任务并在不同设备和不同层次之间进行分配, 而网络协议则试图在任务卸载之后提升不同设备间的通信和协作效率. 不同于传统的视频传输重视人眼观看视频的体验,

视频分析过程中的流式传输更看重最终的分析准确度, 且传输的内容并不只包括视频流. 通过对编码方式、传输内容、压缩程度等进行调整, 网络协议需要在波动和受限制的网络环境下最大化视频流分析效果. 此外, 许多工作提出的网络协议可以与第 4.1.4 小节中的本地过滤方法有机结合, 进一步降低传输带宽.

针对视频流. 许多工作考虑把视频流或视频帧传输到边缘服务器然后进行分析的情况, 试图针对视频内容和视频分析特性设计网络协议, 节约传输带宽.

其中部分工作考虑将整个视频流完整地从设备传输到服务器. AWStream [120] 借鉴了 DASH [168] 等自适应比特率的无状态视频传输协议的思想, 动态调整下一个视频片段的编码质量以应对网络带宽波动, 最大化分析准确度. 首先, AWStream 通过离线分析和在线抽样调整的方法, 得出不同的帧率、比特率等视频编码参数下对应的视频分析准确度, 找到其中取得 Pareto 最优的参数集合. 然后, 系统根据当前传输队列长度和系统反馈的处理延迟等信息实时估计当前网络带宽, 并选择合适的视频编码参数, 自适应地调整接下来传输的视频质量. 在增强现实、行人检测等任务中, AWStream 相比于 TCP 视频传输降低了 97.5%~99% 的延迟, 相比于 UDP 视频传输提升了 45%~88% 的分析准确度, 相比于基于人工规则的通用流数据分析系统 JetStream [121] 减少了 93.3%~95% 的延迟, 且增加了 1%~5% 的准确度. CloudSeg [122] 采用类似 AWStream 的算法进行低分辨率视频传输, 再运行超分辨率模型在云端重建高质量的图像进行分析. 传统的超分辨率模型目的在于使重建的图片和原始图片像素相似. CloudSeg 将原始图片进行视频分割的结果和重建图片进行分割的结果的差作为损失函数, 对已经训练好的超分辨率模型进行参数微调, 从而使之能够针对视频分析任务进行图像重建. 相比于 AWStream, CloudSeg 在同样的分析准确度下进一步减少了实时视频流分析 85.3% 的带宽占用. 其提出的结合超分辨率和语义分割的分析任务可以在服务器按照 23.5 fps 运行.

部分工作则根据服务器的分析反馈, 找到感兴趣区域或视频片段进行重点传输, 从而减少冗余信息对带宽的占用. 其原因在于, 对诸如目标检测和语义分割等许多视频分析任务来说, 原始视频流中不包含目标的区域对视频分析没有任何贡献. 例如, 在车流识别任务中, 监控摄像头拍下的路面和路灯等就属于冗余信息. Vigil [44] 在第 4.1.4 小节中提到的视频帧筛选的基础上, 在系统层次融合各个边缘节点的分析信息, 提出了基于内容的包流量调度和边缘服务器之间带宽分配的机制. 首先, Vigil 根据每个视频帧对应的目标个数进行排序, 优先上传包含目标多的视频段; 然后, Vigil 通过差分轮询算法, 系统按照所有视频流单位时间内检测到的目标数目给边缘节点分配带宽, 最大化云端应用在固定带宽下每秒能够收到的总目标个数. SkyEyes [123] 针对无人机对特定目标进行搜救的场景, 在无人机上检测出目标, 将检测到的所有目标对应的方向梯度直方图特征压缩后实时传给基站, 如果基站发现特征与目标匹配, 再向无人机请求对应的视频片段. 在目标出现概率小于 50% 的情况下, SkyEyes 相比于 H.264 压缩整个视频流能节约更多的带宽. 根据 Pakha 等 [124] 的初步研究, DDS [125] 采取了与 SkyEyes 相似的两路传输做法: 摄像头持续向服务器传输低质量的视频流, 服务器对齐进行第一次分析后, 找到其中可能具有目标, 但是在低清视频中难以分析的区域, 并将其反馈给摄像头. 收到反馈区域后, 摄像头把每一帧中不感兴趣的区域全部置为黑色, 将对应视频片段进行高质量编码, 再次传输给服务器进行二次分析. 由于 H.264, H.265 等视频编码算法优秀的运动补偿机制, DDS 将大部分区域置为黑色的方法可以有效减少高清视频流的带宽占用. 此外, DDS 还通过 Kalman 滤波器结合反馈调节机制预测当前网络状况, 动态调整高/低质量两路视频流的具体压缩参数. DDS 宣称, 在保持同样视频分析准确度的前提下, 比 AWStream, CloudSeg, Vigil 等系统节约了 59% 的传输带宽. 然而在 DDS 中, 部分分析结果需要等到两次视频传输之后才能得到, 在最差情况下的响应延迟在 2 s 以上. 此外, 上述方法的视频大部分采用了分段发送视频流的方式, 如果考虑到等待视频帧组成视频片段的时延,

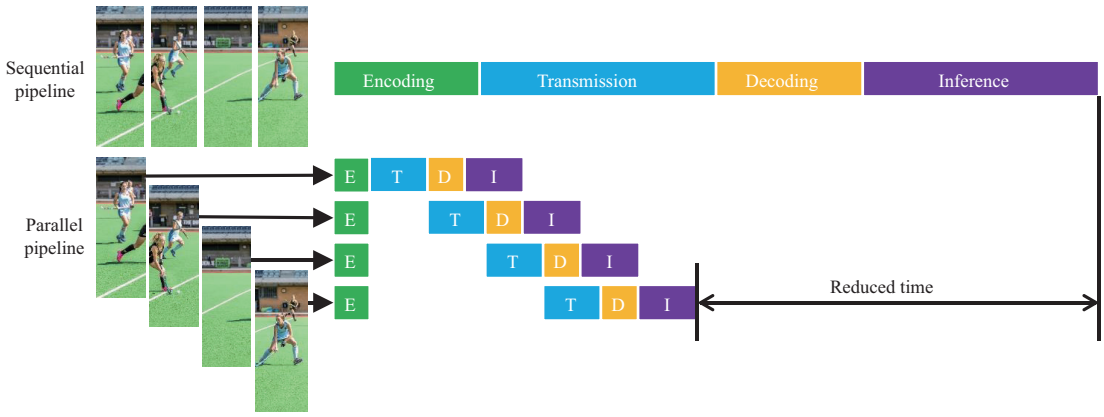

图 17 (网络版彩图) 编码、传输、解码和模型推理并行流水线

Figure 17 (Color online) Pipelined parallelism for encoding, transmission, decoding, and inference

系统的最低理论端到端时延为视频片段的长度, 通常为 2~10 s [120].

为了服务增强现实等高度延迟敏感的任务, 还有部分工作尝试进一步通过网络传输协议降低分析任务的端到端延迟. EAAR [8] 将视频帧按照相同方向切割为 4 份, 按照如图 17 所示的流水线方式, 将视频帧编码、传输、解码、模型推理 4 个步骤并行执行, 减少了每个视频帧的端到端延迟. 注意到, 由于卷积神经网络的特性, 图像切割位置的卷积需要在接收到相邻两个图像块之后再进行计算. 此外, EAAR 还采用了 ROI 编码的方式, 对服务器目标检测模型预测的 ROI 区域及其附近采用高质量编码, 其他区域采用低质量编码. EAAR 最终能达到 60 fps 的分析速度, 并且相比于直接上传图片进行分析提升了 20.2%~34.8% 的准确度. EagleEye [126] 针对在拥挤人群中找到特定人脸这一特定任务, 同样将视频帧分块进行流水线分析. 此外, 对于手机检测出的每一个人脸, EagleEye 根据其清晰度和方向进行划分, 将清晰且易分辨的正脸放在手机 GPU 上进行识别, 将其他人脸卸载到边缘服务器分析, 增加了分析流程的并行程度. 实验表明, EagleEye 的并行分析流水线将端到端延迟减少了 88.9%, 达到了 1 fps 的速度. EdgeDuet [77] 同样采用了 ROI 编码和流水线并行技术, 并注重增加可能存在小目标的部分图像块质量. 为了解决检测目标被分开的问题, EdgeDuet 的图像分块之间包含了固定宽度的重叠区域. 不同于 EAAR 对每个图像块分别进行编码和传输, EdgeDuet 还修改了 H.265 开源压缩算法, 从视频编码层级进行了分块编解码. 相比 EAAR, EdgeDuet 提升了 44.8% 的分析准确度和 223% 的小目标分析准确度, 并且进一步减小了 34.2% 的延迟.

针对特征层. 还有许多边缘实时视频流分析系统采用了模型分割方式进行任务卸载, 需要在网络中传输深度学习模型的特征层而非视频流. 由于卷积层输出的中间特征图体积远大于压缩后的图像或视频帧, 直接将原始特征层在设备和服务器之间传输具有较大的开销. 因此, 部分网络协议考虑在模型分割后利用特殊的编码方式对特征进行压缩, 在保证分析准确度的前提下减小模型分割带来的带宽占用.

Ko 等 [127] 将卷积层放在终端运行, 将全连接层卸载到云端, 并考虑了两种特征层压缩方式. 其中, 无损压缩利用了行程编码 (running length encoding) 和 Huffman 编码结合的方式, 达到了 33%~10% 的压缩率. 有损压缩直接采用了通用的图像压缩格式 JPEG 对特征图进行压缩, 并利用压缩后的特征图作为输入微调边缘服务器端的模型参数, 在损失 1% 分析精度的情况下达到了 20%~2% 的压缩率. 仿真结果表明, Ko 等的特征图压缩编码相比于在边缘服务器直接执行提升了 2.5 倍的吞吐量, 并减少

了 56.5% 的能耗. JALAD [128] 将特征图中的数值从浮点数映射到宽度为 c 个比特的整型变量, 然后用 Huffman 编码进一步对其进行压缩, 达到了 10~100 倍的压缩率. 为了权衡压缩率、网络状况和视频分析质量, JALAD 记录了不同比特宽度 c 对应的模型准确度, 然后将其和模型分割位置一起纳入整数线性规划框架求解, 最小化系统的端到端延迟. 相比于直接传输 JPEG 压缩后的图像到云端进行分析, JALAD 能够减少 90% 的端到端延迟, 但是可能损失 10% 以内分析准确度. 类似地, Emmons 等 [129] 在特征图的量化和 Huffman 编码中间, 加入了离散余弦变换和卡洛南 – 洛伊变换 (Karhunen-Loève transform, KLT) 等信号处理方法进一步进行压缩. Matsubara 等 [130] 在进行模型切割后通过知识蒸馏方法对卸载到设备上的模型进行体积压缩和训练, 并且在模型分割点处引入一层神经网络作为瓶颈再次训练, 减少网络中传输的特征图大小. 该方法在 DenseNet [169] 网络架构上进行模型分割实验, 能够在降低 1% 以内延迟的情况下降低 98% 的网络带宽占用以及 85% 的设备端计算开销. Hu 和 Krishnamachari [131] 先用动态规划算法, 以负载均衡为目的找到当前神经网络的最佳分割点, 然后在任意两个被分割的网络层之间加入编码器 – 解码器结构的卷积神经网络作为压缩器, 并对修改后的整个网络进行重新训练, 端到端地学习压缩器参数. 通过调整编码器 – 解码器的参数, 他们还可以训练出多个不同压缩率的压缩器, 从而根据网络带宽的波动进行动态选择和替换. 将 Hu 和 Krishnamachari 提出的端到端压缩方法应用于 MobileNet-v2, 可以在波动的网络条件下取得 32% 的分析准确度提升. 部分低功耗小型物联网设备相比于智能手机仍少了 2~3 个数量级的计算资源, 而且带宽十分有限. 即使通过纵向分割和特征图压缩, 这些设备较低的无线带宽和有限的电量仍不足以完整传输深度学习模型的中间特征层. 针对这种小型物联网设备, CLIO [108] 充分考虑了 IoT 设备上的计算资源和通信资源, 将模型压缩、纵向分割和自适应传输相结合. 具体来说, CLIO 对神经网络中间层进行分片和渐进式训练, 使得模型仅根据第一个分片即可完成推理, 分片越多模型准确度越高. 在运行时, CLIO 根据不同的网络条件渐进式分片传输中间层, 在计算资源和带宽资源十分有限的情况下, 尽量保证模型推理的性能.

4.2.3 隐私保护

云边端跨层多设备协作进行实时视频流分析, 涉及大量数据的传输和存储, 而这些数据通常包括人脸、车牌等个人隐私. 为了保护数据隐私, 许多视频分析系统在上传视频流或关键信息之前会对原始数据进行脱敏处理. 一方面, 隐私保护技术需要对视频流中的关键部分进行有效剔除或模糊, 避免攻击者复原出隐私信息. 另一方面, 为了支持实时视频流分析, 其还需要考虑对部分通用视频分析任务准确度的影响. 同时, 部分值得信赖的第三方, 例如执法部门, 也应该有权限逆转隐私保护操作, 获得原始视频数据. 此外, 由于云服务器以及公共网络的不安全性, 通常情况下对数据的脱敏处理需要在设备或者边缘服务器上自动实时运行, 需要权衡视频分析和隐私保护任务对有限计算资源的竞争.

目标脱敏. 如图 18 所示, 早期的视频数据保护主要依靠目标检测找到特定类型的目标然后进行模糊或遮挡. GigaSight [132] 提出了一个基于边缘计算的分布式众包视频上传和实时自动标注系统, 并在边缘服务器进行视频脱敏和存储. 该系统允许用户通过规则指定哪些信息为需要保护的, 并在边缘服务器检测出对应目标进行遮挡. 为了权衡系统的吞吐量和脱敏质量, 系统仅对原始视频降低分辨率和帧率进行脱敏处理, 提供给云端进行索引, 而将原始视频加密保存在本地. 其处理速度约为 7.16 fps. Wang 等 [133] 实现了对原始视频流的实时脱敏处理. 他们针对实时人脸识别应用, 利用基于目标检测的追踪方法对已经识别的人脸进行追踪和模糊, 从而减少重复识别. 原始人脸数据被加密保存在本地, 剩余的视频部分和识别结果上传到云端进行进一步分析. 为了提高脱敏准确率, Wang 等设置了长度为 1 s 的待验证缓冲区, 允许应用根据新的检测信息对人脸位置进行更新. Hassan 等 [134] 则先将整个

图 18 (网络版彩图) 目标脱敏示意图
Figure 18 (Color online) Target denaturing

图 19 (网络版彩图) 全局脱敏示意图
Figure 19 (Color online) Global transformation

视频流通过图像处理方式转换为卡通风格, 然后其中部分预设好的敏感类别进一步用数据库中已有的相似图案进行替换. 将敏感部分遮盖或模糊可以避免隐私泄漏, 但是也影响了视频分析效果. 因此, 一些工作将原始数据中的目标替换为生成的同类目标, 可以既保护隐私, 又不影响后续视频流分析. 其中 Gafni 等 [135] 利用对抗学习的方法将视频流中的人脸实时替换为生成的人脸. 他们将一个已有的人脸分类器输出加入生成对抗网络的隐变量中, 并巧妙设计了训练损失函数, 使得生成的人脸与原始人脸具有相同的朝向和表情, 但是完全不同的身份. Bertran 等 [136] 则试图通过具体分析任务指导对抗学习训练, 使得生成的图像能够让特定视频分析任务无效, 同时不影响另外一个特定任务. 例如, 在避免情绪识别的前提下保留人脸的性别特征, 或在保持身份验证特性的同时欺骗性别识别算法. 基于目标生成和替换的方法通常能够较好完成隐私保护和视频分析的权衡, 但是具有计算复杂度高的特点, 当场景中人脸较多时无法实时运行, 只能离线处理视频数据.

全局脱敏. 隐私保护具有个性化特点, 不同的人可能对于何种目标是敏感信息具有不同的认知, 因此基于针对特定目标进行脱敏的方法难以满足所有用户的需求. 如图 19 所示, 有的工作尝试对整个视频帧进行全局变换脱敏处理, 并努力克服该操作对通用视频分析带来的影响. Raval 等 [137] 利用自动编码器模型的生成对抗网络对图像进行处理, 试图在图像脱敏的同时不影响图像分析相关的特征. 其中, 判别器模型试图判断重建的图像是否与公开数据集的图像相似; 生成器模型则试图生成出与公开数据集图像相似, 并且保留原始图像分析特征的新图像. 实验表明, 利用参数微调后的图像分类器在重建的图像上进行分类能够保持 91% 的准确率. 此外, Raval 等故意在部分输入图像中加入了公开数据中不存在的二维码图案, 发现系统生成的图像完全不包含二维码. ARDEN [138] 针对基于模型分割的视频流分析系统, 指出神经网络的特征图有可能被恶意恢复成原始图像, 泄漏用户隐私. 因此, ARDEN 对本地模型提取的特征图进行了随机擦除和噪声添加, 并对云端模型通过原始和生成数据进行针对性训练以维持分析准确度. 在 CIFAR-10 的实验中, ARDEN 相比没有脱敏的方法仅下降了 7.9% 的分析准确度. PECAM [139] 利用生成对抗网络 CycleGAN [170] 的变体进行视频流实时脱敏处理, 试图同时做到全局隐私保护和不影响通用计算机视觉算法的分析, 并且允许授权第三方利用包含密钥的特定模型恢复原始视频. 具体来说, PECAM 将摄像头对应的二维密码作为输入图像的透明度层, 并训练生成对抗网络生成如图 19 所示的卡通风格图像. 在部署时, 生成对抗模型中的转换器 (transformer) 被用于对视频帧进行脱敏处理; 重建器 (reconstructor) 被用于和摄像头的二维密码一起将脱敏后的视频帧进行还原. 此外, 为了在视频编码效率和可恢复性之间进行权衡, PECAM 对生成视频帧中信息熵较高的像素块进行无损编码, 其他部分进行有损编码, 与 H.264 解码器兼容的同时提升了 1.8 倍压缩

效率. 实验表明, PECAM 脱敏后的视频能在多种视频分析中相对原始视频保持 96% 的准确度, 同时隐藏了人物身份、车牌号码等敏感信息.

4.2.4 小结

云边端跨层协作, 以及同层的多设备协作, 是边缘计算范式的重要特征, 其主要目的是在复杂的设备 – 边缘服务器 – 云端通信条件, 以及异构的计算资源下解决视频分析任务的相互卸载问题, 并处理好设备间的协作通信. 其中任务卸载主要考虑计算任务的划分以及资源分配; 网络协议和隐私保护则关注设备间协作通信的效率和信息保护. 跨设备任务协作和资源分配为边缘实时视频分析带来了网络、服务器资源等更加丰富的优化维度. 不同于终端设备层次技术必须在分析效果和性能间进行取舍, 协作层次技术能够同时提升系统的分析延迟和准确度. 我们将涉及协作层次技术的部分代表性工作的应用场景、核心方法和优缺点总结在了表 3 中. 可以发现, 由于涉及多设备合作, 具有比较丰富的调节空间, 每个协作层次技术通常可以同时优化视频分析的多个目标. 而且, 其中许多技术都针对实际任务的特点专门设计了设备间协作方式.

具体来说, 任务卸载技术按照视频分析模块或深度学习模型粒度对计算任务进行卸载, 能够利用远大于单个设备的计算资源更好地完成实时视频分析. 其中, 纵向卸载利用了边缘/云服务器相对较强的计算能力; 横向卸载则利用了周围设备的空闲计算资源. 针对实时视频分析的边缘任务卸载方法已经经过了广泛的研究, 在各项分析任务中应用广泛. 然而, 任务卸载需要传输视频源或中间计算结果, 对网络带宽和延迟的要求较高. 即使部分方法将网络状况纳入任务划分的限制条件, 网络的突发波动仍然对视频分析效果有较大负面影响. 网络协议则专注于计算任务卸载之后对不同设备间的通信和协作效率进行优化. 通过对数据流特性和应用需求的针对性传输方式设计, 网络协议试图最大化网络波动或有限带宽下视频分析的最终效果. 针对增强现实和可穿戴认知辅助这类人机交互应用, 网络协议包含了流水线、渐进传输等低延迟技术; 针对交通和安防监控等规模较大的应用, 网络协议则可以利用编码和压缩技术降低单个视频流的带宽占用. 如果不需要任务卸载仅在移动端进行视频分析, 或移动端与边缘服务器通过内网连接, 则不需要单独进行隐私保护. 反之, 若是要通过互联网这类不可靠的信道进行任务卸载, 高效且可靠的隐私保护技术则具有重要作用. 不论是局部或是全局脱敏, 隐私保护技术通常都需要额外的视频分析, 因此会为系统带来一定的计算负担. 同时, 其应用也可能影响部分不涉及隐私的其他视频分析任务的效果.

4.3 边缘/云层次

边缘/云层次包括了查询优化、推理加速和边缘缓存等技术.

4.3.1 查询优化

查询优化 (query optimization) 原本是数据库领域的术语, 指数据库在执行用户复杂的查询请求时, 考虑等价替换、去重、并行执行、存储和计算资源分配等多种优化方法, 自动估算满足查询的不同方法的执行成本, 进而确定最优查询计划 (query plan) [140]. 在大规模的实时视频流数据分析中, 一个边缘/云服务器往往会服务多路视频, 同时多个用户可能向服务器发出不同的请求, 例如实时计算当前视频中的所有车辆位置、在区域内所有视频流中搜索某个目标人物的位置、判断视频中的车辆类型和提取车牌号码等. 这些请求同样可以被视为用户向服务器发送的查询请求, 而需要执行的查询计划就由第 2.1 小节提到的视频分析流程组成, 涉及了各种图像处理算法和深度学习模型的执行. 大部分查询优化技术按照模块为单位进行优化, 将视频分析流程中的深度学习模型作为黑盒模块, 通过外部测

表 3 协作层次技术相关文献总结对比

Table 3 Summary and comparison of selected studies of collaboration level technologies

Ref.	Application	Key method(s)	Pros.	Cons.
[52]	AR	Runtime offloading decision making	• Measurement-driven optimization framework • Adapt to network dynamics	• Fixed offloading method • Single tenancy
[17]	Surveillance	Cross device-edge-cloud resource allocation	• Bandwidth & computation co-optimization	• Ignore latency constraints
[99]	AR	Region proposal prediction; multi-server offloading	• Reduce latency & bandwidth • Load balancing	• Experiment on simulation
[100]	General	Layer-level model partition	• Reduce latency & energy	• Only for sequential models
[103]	Surveillance	Model partition with early exit	• Reduce communication overhead	• Handcrafted partition point
[104]	General	Joint partition & exit point selection	• Adapt to latency constraints & network dynamics	• Pre-defined exit point
[105]	General	Synergistic progressive inference	• Adapt to resources & input • Can run without network	• High power consumption
[107]	WCA	Incremental upload of DNN; partial DNN inference	• Suitable for cyber foraging • Reduce overall latency	• Cannot support RNN
[108]	General	Feature map slicing; progressive offloading	• Adapt to network dynamics • Reduce power consumption	• Only focus on image tasks
[109]	General	DAG-structure DNN partitioning	• Max-flow min-cut formulation • Adapt to network condition	–
[112]	AR	Cross device load balancing; response time estimation	• Improve inference speed & throughput	• Experiment on synthetic data
[113]	Surveillance	Runtime horizontal offloading; camera specific DNN finetune	• Improve accuracy • Save bandwidth	• One-time offline finetune
[116]	Surveillance	LSTM-based ROI prediction; stochastic partitioning	• Reduce average latency • Improve resource utilization	• Experiment with simple tasks
[117]	Detection	MapReduce-based distributed DNN inference	• Reduce latency • Adapt to device resources	• High bandwidth & energy demand
[118]	Detection	Fused tile CNN partition; runtime load balancing	• Reduce communication overhead • Open source	• High power consumption
[120]	AR	Adaptive bitrate streaming analytics	• Tradeoff on accuracy and bandwidth • Adapt to network dynamics	• Lack fault tolerant • Exhaustive profiling
[122]	General	Super resolution-based streaming analytics	• Accuracy-aware SR • Reduce bandwidth	• High computation overhead
[125]	General	Two-fold server driven streaming	• Reduce more bandwidth	• High worst-case latency • 2x computation overhead
[8]	AR	Dynamic ROI encoding; pipelined offloading	• Reduce transmission size & latency	• Ignore network dynamics
[77]	AR	Content prioritized tile offloading	• Improve small object detection • Specialized video encoder	–

续表

Ref.	Application	Key method(s)	Pros.	Cons.
[127]	Classification	JPEG feature map compression	• Improve throughput • Reduce energy consumption	• No accuracy guarantee
[131]	General	Autoencoder-based feature map compression	• Adapt to network dynamics • Improve accuracy • Save bandwidth	• High training overhead
[132]	Video tagging	Face detection & denature	• Tradeoff on throughput & privacy	• Cannot run in real-time
[135]	Recognition	Real-time face replacement by GAN	• Denature face identity • Keep expression & orientation	• High computation overhead
[138]	General	Perturbed feature map; noisy training	• Avoid image recovery	–
[139]	Surveillance	Privacy-enhanced reversible video transformation	• Perform real-time on camera • Maintain analytics accuracy	• No theoretical guarantee against attack

试得到其属性, 然后分别对每个模块分别进行资源分配、模块调度, 以及参数优化. 总的来说, 查询优化技术试图在边缘/云服务器上, 对每个查询请求的分析准确度、查询延迟、资源占用等指标进行权衡, 并最大化服务器的服务规模.

Optasia [141] 对数据流处理引擎 SCOPE [171] 进行了扩展, 使之支持多种基础的视频分析模块. 具体来说, Optasia 将特征提取、分类、检测等单帧操作作为处理算子 (processor); 将背景去除、目标追踪等连续视频流操作作为规约算子 (reducer); 将目标重识别等跨视频流操作作为合并算子 (combiner) 整合进了 SCOPE 系统中, 实现了类结构化查询语言 (structured query language, SQL) 接口. 该用户接口可以用 10 行以内的代码实现安铂警报、交通违法检测、行人重识别等任务. 为了增加并行执行的机会, Optasia 将连续视频流切分为多个相互重叠的视频片段, 对整个视频流的持续操作变成多个对这些片段的独立操作. 此外, 在系统已有的谓词下推、子表达式合并、有向无环图并行执行等查询优化方式基础上, 针对视频分析提供了一系列表达式等价替换规则, 在运行时根据完成时间选择其中最优的执行方式. 实验表明, Optasia 提供的特殊优化方式能减少 1.5~3 倍延迟, 并且具有良好的扩展性, 在输入规模提升 100 倍后查询延迟几乎不变. VideoStorm [15] 则进一步考虑了视频分析流程中多种可配置的参数, 包括帧率、分辨率、模型压缩等, 对执行同一个查询请求的准确度、计算开销带来的权衡. 与普通 SQL 查询基于分析模型预测执行开销不同, VideoStorm 通过离线测试分析不同参数配置下的分析质量和 CPU 占用, 通过启发式搜索找到其中达到 Pareto 最优的参数集合. 在处理实时视频请求时, VideoStorm 将上述选出的参数配置集合作为优化问题的解空间, 找出当前集群 CPU 资源限制下最小化平均延迟和最大化分析准确度的查询计划, 并进行相应的资源分配和任务调度. 如图 20 所示, VideoStorm 根据确定的查询计划, 按照负载均衡和提高利用率原则, 将视频分析流程中的各个子模块分配到不同机器上执行. 在 Azure 上 101 个机器的集群部署实验中, VideoStorm 相比于基于公平分配的调度方式提升了 80% 的分析准确度, 并且超过延迟要求的查询数量降低了 85.7%. 作为 VideoStorm 的后续工作, 第 4.2.1 小节中提到的 VideoEdge [17] 采用了类似的分析质量 – 资源权衡思路, 但将优化范围扩展到了设备 – 边缘 – 云端三级架构, 考虑了网络带宽和不同层级服务器的 CPU 个数等多种资

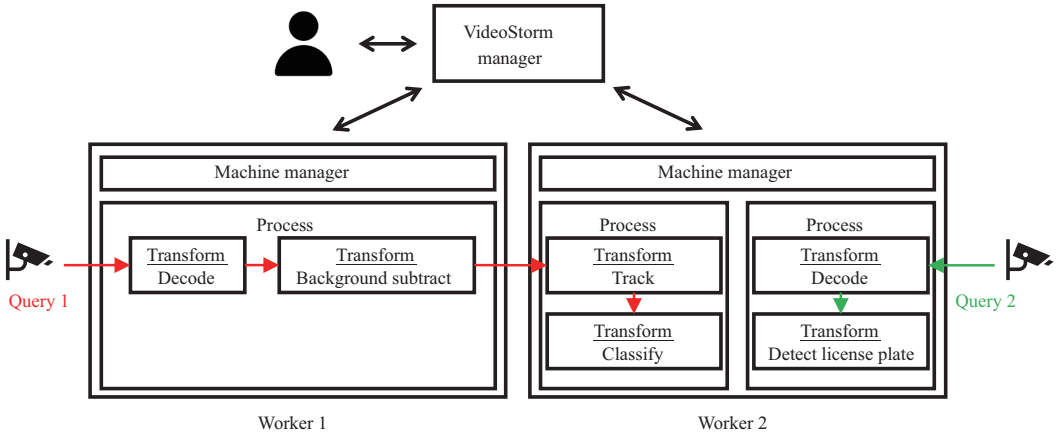

图 20 (网络版彩图) VideoStorm 系统架构示意图

Figure 20 (Color online) VideoStorm system architecture

源的限制, 以及不同视频分析任务共享相同底层任务结果的可能性. VideoEdge 相比 VideoStorm 提升了 5.4 倍的分析准确度.

上述方法采用一次性离线分析得出视频分析参数对应的准确度和资源占用, 并且没有考虑 GPU 等异构计算资源. Chameleon [142] 指出, 视频分析流程最好的参数配置会随着视频的内容发生变化, 因此在运行时周期性进行参数分析, 选择最适合当前视频的最佳参数组合. 为了缓解周期性分析带来的计算开销, Chameleon 利用了视频流的时间局部性和部分视频流之间的相关性, 在一定时间范围和一组摄像头内, 共享最佳的 k 个参数配置. 相比于仅离线进行一次参数分析, Chameleon 能在相同的计算资源下提升 20%~50% 的分析准确度. HeteroEdge [143] 针对边缘服务器上异构的计算设备和边缘服务器集群间的网络状况, 将视频分析流程中的模块映到 Storm 框架中作为子任务进行调度. HeteroEdge 根据对每个子任务的计算延迟和资源需求进行估计, 将每个任务调度到处理延迟加网络传输延迟最小的服务器执行, 并将特定瓶颈任务放在 GPU 上执行. 相比于 Storm 框架的现有调度算法, HeteroEdge 在同时具有 CPU 和 GPU 的服务器集群上减少了 40% 的视频帧平均处理时延.

4.3.2 推理加速

在大规模视频流分析过程中, 深度神经网络是提高吞吐量的瓶颈所在, 占据了服务器的大部分计算能力, 并且通常需要在 GPU 等专用硬件上加速执行. 查询优化技术按照黑盒处理各个视频分析模块, 并且根据不同请求分别独立进行资源分配, 通常没有考虑深度神经网络和加速设备的硬件特性. 而服务器上的推理加速技术则针对深度神经网络推理和硬件特性进行白盒优化, 进一步提升计算资源的利用效率. 推理加速可以分为输入合并和参数合并两个维度: 第一, 当多个视频流采用了相同的神经网络进行分析时, 可以采用自适应批处理方式合并模型输入, 提高设备的资源利用效率; 第二, 同一个和多个视频流利用多个不同模型进行多种分析时, 可以通过共享部分神经网络子结构的方式合并模型参数, 减少内存开销和总计算量. 基于以上两个优化方向, 模型合并需要在保持分析准确度和满足每个应用延迟限制的情况下, 尽可能提高单位计算资源下的系统吞吐量.

输入合并 (批处理). GPU 有多个并行计算核心和极高的带宽. 通过批处理方式同时合并处理多个输入, 可以有效提升神经网络对 GPU 资源的利用效率, 增加系统吞吐量. 然而, 批处理方式也会增加单个分析请求的响应延迟: 一方面, 部分输入需要等待其他输入组成批次; 另一方面, 批次大

小的增加也会增加一次模型推理的执行时间. TensorFlow serving [144] 是谷歌公司针对 Tensorflow 深度学习框架开源的模型服务系统, 利用批处理自动合并所有相关输入以支持大规模模型推理. 然而, TensorFlow serving 的目的主要在于提供接口简化深度学习模块的工业级部署, 需要用户手动设置批次大小. Clipper [145] 则通过递增批大小然后指数回退的方式, 快速测试出给定延迟要求下当前硬件的最大批次大小, 并将其发送到对应模型的容器进行分析. 在突发负载或较少负载情况下, 可能出现部分批次输入数量不到最大批大小的情况, 此时 Clipper 将延迟执行当前批次以等待更多输入. 在部分数据集中, 延迟 2 ms 执行批处理可以提高 3 倍系统吞吐量. 此外, Clipper 还通过模型结果的缓存和复用减少重复计算, 并且在线模型选择和集成学习技术, 利用多个模型的输出提升分析准确度. 实验表明, 通过自适应批处理和输出结果缓存, Clipper 可以提升 26 倍系统吞吐量, 与包含大量硬件优化的 TensorFlow serving 表现相当, 并且基本满足应用的延迟要求. 但是, 在 Clipper 提出的多模型集成批处理框架中, 部分模型运行速度较慢, 其落后效应会影响最差情况下的系统端到端延迟. 因此, 为了减少模型推理延迟的波动性, CollageInference [146] 提出了拼贴推断技术, 将一组输入图像拼接为一张图像放入定制的网络中计算出每个输入的粗略结果作为冗余备份, 当部分模型推理速度过慢时, 即可直接使用拼贴推断解码出的对应结果. 该方法仅针对图像分类任务, 可以减少 16.7%~50% 的 99 百分位延迟, 并且降低 33.3%~44.4% 的延迟方差. 不同于 Clipper 运行在单个 GPU 的服务器上, Neuxs [147] 考虑了具有多个相同 GPU 的集群, 提出了挤压装箱问题 (squishy bin packing). 具体来说, 他们假设每个 GPU 中的所有模型按照轮询调度执行, 然后根据批处理能增大吞吐量和减小处理延迟的特性, 给每个 GPU 分配模型和确定模型最大的批次大小, 在满足 99% 延迟要求的情况下最大化系统吞吐量. 此外, 他们还利用了类似 Clipper 的算法进行批次大小动态调整. Neuxs 相比于基于贪心算法分配 GPU 的比较基准, 在时间敏感的游戏分析任务中提升了 9.4 倍吞吐量, 在时间相对不敏感的交通分析中提升了 1.79~4.32 倍吞吐量, 并且能根据负载动态调整资源利用率. Clipper 的续作 InferLine [148] 进一步考虑了云服务器上具有可以自动扩张的不同 GPU 和 FPGA 等异构设备的场景, 试图最小化深度学习模型批处理的硬件开销. InferLine 通过离线测试得出不同模型在不同设备上的延迟与批次大小关系, 然后用低频规划模块找到资源开销最小且满足延迟要求的硬件配置, 利用高频调整模块根据对任务流量的预测动态调整资源数量以确保延迟. 根据亚马逊云服务器的价格, InferLine 相比 Clipper 在相同吞吐量下减少了 86.8% 的硬件开销, 且能满足 99% 的延迟要求.

除了上述面向数据中心的大规模批处理方法, 还有一些工作专门面向边缘环境的输入合并和优化. 针对多个边缘计算节点间的任务卸载, EdgeBatch [149] 试图在保证边缘节点 GPU 并行计算能力充分利用的同时, 权衡时延和系统能耗. 具体来说, 他们分别定义了时延损失函数和能耗损失函数, 将批次大小决策抽象为一个带约束的优化问题进行求解, 通过离线评估和在线学习两个阶段动态调整批次大小. 即使需要处理突发的神经网络计算任务, EdgeBatch 仍然可以在复杂网络环境下实时决定边缘智能系统中最佳的 GPU 批处理策略. 在车牌识别任务中, EdgeBatch 比其他算法降低了 31.1% 的端到端延迟, 并且显著地节约了系统能耗. Fang 等 [150] 针对多租户边缘服务器提出了基于启发式算法和基于深度强化学习的两种 GPU 调度方法. 这些方法可以动态地选择最合适的神经网络模型和批次大小, 使系统在精度、延迟以及响应时间这几个指标上达到最佳的服务性能. 有些应用虽然对视频分析的响应延迟具有要求, 但是允许任务偶尔错过处理截止时间. DeepRT [151] 针对这种软实时的多租户的场景设计了名为 DisBatcher 的批处理机制. 他们将时间划分为长度相同的连续时间片, 同类型的请求如果落入同一个时间片, 便可以在这个时间片结束的时候开始执行批处理. 这保证了在下一个时间片结束前, 当前的批处理任务已经执行结束. DisBatcher 机制可以在满足各个请求给定的延迟要求的前提下尽可能增加批次大小, 有效提升边缘服务器的吞吐量. DeepRT 主要适用于单个 GPU 的边缘服务

器, 而 ECML[152] 则考虑了多 GPU 资源的场景. 在这种场景下, 降低延迟并提升 GPU 集群的整体吞吐量成为了其首要目标. ECML 使用了 GSLICE[172] 提出的基于自学习的自适应批处理算法. 该算法使 GPU 集群在满足模型推理延迟要求的前提下尽可能地提升批处理大小, 从而获得了更高的吞吐量. 同时, ECML 将单个 GPU 的受控空间复用模式 (controlled spatial multiplexing) 扩展到了多 GPU 集群上. 其修改了 GPU 执行完任务后的通知机制, 从而减小了回调带来的延迟, 并通过在 GPU 集群范围进行空间共享来服务多个不同的应用程序. 实验表明, 在满足不同任务的低延迟要求前提下, ECML 提升了 2 倍以上的总吞吐量.

模型合并. 许多任务需要对同一个视频流进行不同分析, 例如对同一监控视频流进行人脸识别、车辆检测和车牌识别等. 由于迁移学习的广泛应用, 许多深度神经网络具有相同的网络架构和相近的参数, 有的工作考虑将这些不同网络架构直接或者经过参数微调之后进行合并, 既可以降低模型加载和内核启动的开销, 也可以减少计算量. 此外, 迁移学习过程中微调的参数越多, 神经网络对特定任务的效果越好, 因此模型间参数合并涉及分析准确度和吞吐量之间的权衡. Mainstream[153] 针对同一个视频流进行多种不同场景的分析, 有两个主要模块. 其中训练模块将多个模型进行联合训练微调参数, 使之具有相同参数, 并在训练过程保留次优模型及参数; 调度模块根据训练模块的信息和在目标机器上逐层测试的性能信息, 在运行时动态调整各个模型之间的参数共享程度, 在满足延迟要求的前提下最大化平均分析准确度. 相比于最大化参数共享和不进行参数共享, Mainsteram 分别提高了 47% 和 87% 的分析 F1 分数. HiveMind[154] 把多模型分析视作对模型参数进行批处理, 对同一个视频流的不同模型进行分组, 将组内的模型进行合并. 具体来说, HiveMind 提出了跨模型层合并技术, 可以将具有相同参数的模型层、具有相同输入数据和输出大小的模型层, 以及具有相同输入大小并且无状态依赖的模型层这 3 类进行合并. HiveMind 通过启发式搜索对模型进行合并, 并且通过输入预处理等方式进一步减少数据读写和模型载入带来的延迟. 相比于分别并行执行所有模型, HiveMind 提升了 5.7∼10 倍的吞吐量, 并减少了 37.5%∼87.5% 的延迟方差. NetFuse[155] 则考虑了不同输入且参数不同的模型进行合并, 通过预设的规则将具有相同结构的模型进行等价合并, 例如将卷积替换成组卷积, 将层归一化替换为组归一化等. 由于对 GPU 内存和缓存资源的高效利用, NetFuse 将经典的视频分析任务最高加速了 3.6 倍.

4.3.3 边缘缓存

除了第 4.1.3 小节提到的单个视频流本身具有局部性特点之外, 在相邻地点、相近时间的不同视频分析请求也具有一定的相似性. 例如, 在超市中可能有很多用户利用手机识别商品信息, 同一个位置进行增强现实游戏的用户可能需要在相似的目标上进行渲染. 边缘计算服务器位于用户的物理位置附近, 天然地需要处理相邻地点、多个用户的相似请求. 因此, 边缘缓存技术以一定的存储开销为代价, 可以通过合理复用分析结果减少冗余计算量和边缘 – 云端通信, 并且提高平均处理延迟. 图 21 为实时视频分析中通用的边缘缓存架构. 边缘服务器通过编码器对图像建立索引和缓存分析结果, 仅在缓存未命中情况下请求云服务器进行神经网络分析.

Cachier[157] 针对增强现实所需的实时目标识别场景, 按照最近最少使用的替换策略, 在边缘服务器对识别结果进行缓存. 当用户发送图像到边缘端进行识别时, 边缘服务器计算其二元 ORB 特征[173] 并在缓存中通过位置敏感哈希 (locality sensitive hashing) 寻找最佳匹配, 如果匹配成功则复用识别结果, 否则将特征发送到云端进行进一步识别. 为了最小化用户请求的平均处理延迟, Cachier 利用最大后验估计模型预测缓存命中率, 并根据网络状况动态调整边缘端缓存的大小. 根据仿真实验, Cachier 相比于全部上传到云端分析减少了 66.7% 的平均处理延迟. Cachier 的后续工作 Precog[158] 提出了设

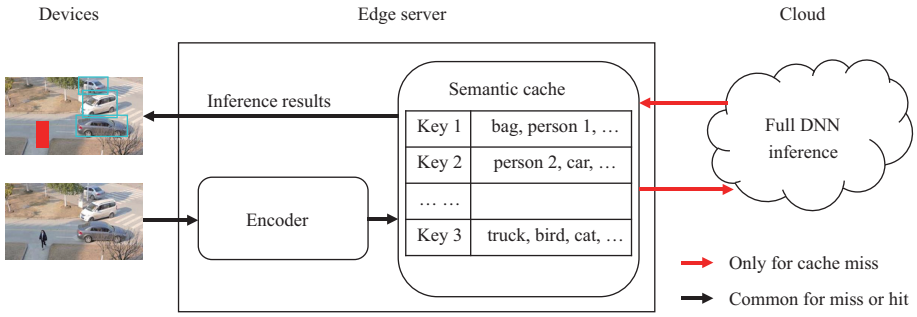

图 21　(网络版彩图) 边缘缓存系统示意图

Figure 21　(Color online) Edge caching system

备端和边缘端两级缓存架构, 进一步减少了平均处理延迟. 具体来说, 边缘服务器根据众多用户的识别请求顺序, 通过马尔可夫模型预测当前设备接下来最可能请求的目标. 设备端则根据边缘服务器的预测和本地动态选择的缓存大小提前下载对应的目标特征和识别结果. 相比于只有边缘服务器一级缓存, Precog 在真实数据的实验中减少了 80% 的平均延迟.

上述方法通过简单的底层图像特征进行匹配, 很多相似但不完全相同的图像请求无法很好地利用缓存结果. ShadowPuppetes [156] 针对分类任务, 提出用自动编码器结构的小型神经网络快速计算图像语义特征, 使得类别、身份相似的目标的特征更为接近, 并以该特征值为索引进行缓存匹配. ShadowPuppetes 在各个类别中相比于云端直接识别减少了 80%~90% 的延迟. FoggyCache [159] 采用了类似 Cachier 的架构, 但试图找到一组与当前输入相似的特征, 并综合这一组特征对应的结果得到当前输入的近似结果. 首先, FoggyCache 对传统的位置敏感哈希方法进行了改进, 动态调整算法中的位置敏感参数, 以适应随着时间变化的特征数据分布. 其次, FoggyCache 对位置敏感哈希搜索出的一组结果进行统计分析, 判断其同质性, 如果结果间分歧低于阈值则复用频率最大的结果. 该算法可以用于对图像分类、人脸识别、目标检测等多种任务进行加速, 视频流分析实验表明其相比本地运行和没有缓存的卸载都减小了 66.7%~90% 的延迟, 仅损失了 1% 的准确度.

除了利用输入图像请求的计算机视觉信息之外, 还有一些针对增强现实的工作利用其他的物理世界信息辅助边缘服务器的缓存查询. CollabAR [160] 利用谷歌的 ARCore 框架估计不同相机的位姿, 并比对当前视频帧和最近缓存的视频帧是否覆盖了相同的三维世界锚点, 找到所有相关的缓存视频帧. 然后, CollabAR 通过集成学习的方法将相关视频帧的识别结果, 与当前帧的分析结果加总作为当前帧结果. 该方法能够在商用手机上按照 30 fps 的速度进行实时目标识别, 且由于集成了不同视角的分析结果, 在图像失真的环境下提升了 16% 以上的分析准确度. Younis 等 [161] 则在保存缓存图像时加入粗略的位置索引, 通过手机的 GPS 定位信息先进行一次筛选, 再在位置相近的缓存图像中进行特征匹配和结果复用.

4.3.4　小结

边缘/云层次技术主要是为了提高服务器资源的利用率, 尽量扩大实时视频流分析任务的服务规模. 由于实时视频分析任务具有延迟要求, 边缘/云层次技术还需要对分析准确度、处理延迟、吞吐量进行权衡. 其中, 查询优化和模型合并从服务器资源分配及分析任务调度的角度提高硬件计算效率, 边缘缓存技术则从减少冗余计算的角度提高系统吞吐量. 一方面, 基于边缘站点模式的边缘视频分析系统中, 边缘服务器通常拥有较大的计算能力和集群规模; 另一方面, 许多实时视频分析系统涉及云端

和边缘端的协作计算. 因此许多原本为云计算范式设计的视频流分析和服务框架, 例如 Optasia [141], InferLine [148] 等, 也可以直接应用到基于边缘计算的分析系统中. 作为边缘计算系统中计算和存储资源最丰富的设备, 边缘/云服务器除了满足某个应用的需求之外, 更重要的任务是对多租户进行支持和提升系统吞吐量. 我们将涉及边缘/云层次技术的部分代表性工作的应用场景、核心方法和优缺点总结在了表 4 中. 可以看到, 其中查询优化和推理加速更加重视系统整体吞吐量; 而边缘缓存技术对延迟和准确性的提升更为关注.

具体来说, 查询优化技术根据系统的各种资源情况, 对不同视频分析请求进行资源配置、模块调度和参数优化. 其目的是在具有多个租户的服务器上最大化系统能同时服务的总请求个数, 并尽可能满足每个请求对视频分析准确度、延迟等指标的要求. 查询优化技术适用于图 4(b) 所示边缘站点范式的计算架构, 能够在边缘集群上高效处理智慧零售、安防监控、工业视觉等大规模视频分析任务. 然而, 此技术依赖于对神经网络在特定配置下的分析准确度和运行速度的预测. 对该关系的统计分析不仅需要较大计算开销, 还无法保证准确性. 推理加速技术则将不同视频分析请求统一考虑, 利用 GPU 资源加速神经网络推理平均速度, 缓解服务器性能瓶颈. 其中, 输入合并通过批处理方式充分利用 GPU 的高并行性能, 以部分请求的延迟为代价提升系统吞吐量; 模型合并则通过不同任务间共用模型参数降低内存访问的开销. 推理加速技术既适用于规模较大的计算集群, 也可利用单个 GPU 的个人电脑提供分析服务. 其劣势在于对分析请求的确定性时延保证较差, 仅能优化平均延迟或尽力服务延迟敏感任务. 边缘缓存技术对视频分析结果构建索引从而进行复用, 仅当遇到新的请求时才进一步进行分析. 其利用了同一视频流内容的时间冗余和不同租户分析请求的空间冗余, 通过存储开销减少分析所需计算量. 该方法适合增强现实导游等交互应用和图 4(a) 所示游牧服务计算架构, 一个边缘服务器可以服务并缓存附近不同租户的请求. 其缺点是缓存命中率依赖于用户请求间的相关性, 系统泛化性较差.

5 Argus 视频大数据智能分析平台

基于对实时视频分析和边缘计算技术的深入研究, 我们提出了如图 22 所示基于边缘计算的视频大数据智能分析平台 Argus. 我们将第 4 节所介绍的设备、协作、边缘/云 3 个层次多项核心技术进行了有机整合, 提供了良好且易于扩展的系统设计, 并面向实际应用完善了日志分析、数据存储和挖掘、多模态数据融合、异构通信协议支持等一系列扩展功能. Argus 将所有的功能模块都按照微服务架构进行了解耦, 能够通过容器编排框架快速部署到终端设备、边缘服务器和云端. 并且, Argus 可以基于系统运行情况动态调整资源分配和各模块参数配置. 我们将其针对油田生产环境和应用需求进行了定制, 支持了智能安防、资产监控、环境监测等一系列工业应用. 以视频分析作为核心, Argus 从底层硬件到智能分析再到用户界面, 为拥有 76 口油井、总面积近 900 平方公里的伊拉克玛吉努油田提供了基于边缘计算的一体化智慧油田解决方案. 目前, 其同时服务了 500 余路实时视频流.

Argus 平台主要试图解决大规模视频流分析场景中的实时性、可扩展性和普适性这 3 个关键技术问题:

- 实时性问题是大规模视频流分析的核心. 如第 1 节所介绍, 视频分析系统的处理能力必须超过最新的视频数据产生速率, 并且满足具体应用的延迟要求. Argus 通过分布式边缘计算架构应对实时性问题, 由视频源附近的边缘服务集群对实时视频流内容进行分析和存储, 降低了视频流传输带来的延迟. 同时, Argus 还通过 GPU 加速、自适应批次大小、查询优化等技术进一步提升视频分析的效率.
- 可扩展性指大规模视频分析系统应该具备良好的横向扩展能力, 能够在不改变整体架构和基础设施的情况下, 通过增加计算服务器数量提升系统吞吐量. Argus 平台通过两点设计应对可扩展性问题.

表 4 边缘/云层次技术相关文献总结对比

Table 4 Summary and comparison of selected studies of edge/cloud level technologies

Ref.	Application	Key method(s)	Pros.	Cons.
[141]	Traffic	Chunk level video stream parallelism	• Improve accuracy • High scalability	• Ignore network transmission overhead
[15]	Surveillance	Dynamic config & placement for analytics pipeline	• Tradeoff on accuracy, throughput, and worst-case latency	• Ignore network bandwidth • One-time model profiling
[17]	Surveillance	Cloud-edge-device analytics pipeline placement	• Consider network bandwidth and heterogeneous resources	• Not suitable for GPU
[142]	Surveillance	Periodical profiling & pipeline config update	• Adapt to video content • Utilize camera spatio-temporal correlation	• Naive grouping algorithm
[143]	AR	Cross device module scheduling	• Utilize CPU & GPU	• Heuristic scheduling
[145]	General	Adaptive batching & approximate DNN caching	• Improve 26× throughput • Satisfy application SLA	• Only work on single GPU
[146]	Surveillance	Multi-image collage inference	• Reduce latency variance	–
[147]	Traffic, Game	Batch-aware squishy bin packing; prefix batching	• Scalable on GPU cluster • Maximize throughput	• Work poorly on latency-sensitive tasks
[148]	General	Batch size & accuracy tradeoff	• Auto-scalable on heterogeneous devices	• One-time model profiling
[149]	Traffic	Stochastic optimal task batching	• Reduce latency & energy cost • Consider network dynamics	• Only support single tenant
[150]	Detection	Heuristic & RL-based adaptive batching	• Reduce resource contention	• Waiting in queue causes sub-second level latency
[151]	AR, Self-driving	Admission control & "DisBatcher" mechanism	• Reduce deadline miss rate • Increase GPU utilization	• Only work on single GPU
[152]	General	Controlled spatial-multiplexing; self-learning adaptive batching	• Design for GPU cluster • Improve 2× throughput	–
[153]	General	Dynamic DNN sharing among applications	• Tradeoff on accuracy, latency, and throughput	• Computation overhead for training
[154]	General	Cross model DNN layer fusion	• Improve 10× throughput	• Single input multi-DNN scenario
[155]	General	Cross model DNN operation fusion	• Improve GPU utilization	• Handcrafted fusion rules
[157]	AR	Edge caching with LSH image matching	• Reduce average latency • Adapt to network dynamics	• Need static underlying request distribution
[156]	Surveillance	Autoencoder-based cache index	• Improve cache hit rate	• High computation overhead
[159]	AR	Adaptive LSH & KNN-based image matching	• Aggregate multiple results • Improve accuracy	–
[160]	AR	3D anchor-based frame caching	• Improve accuracy for distorted images	• Evaluation on synthesized distortion only
[161]	AR	GPS-assisted image caching	• Reduce image matching overhead	• Require additional sensor data

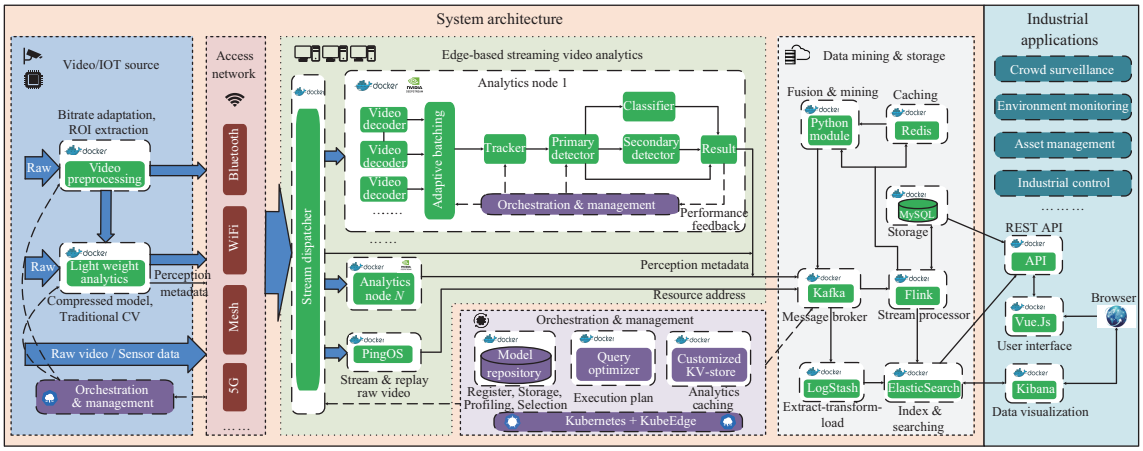

图 22　Argus 视频大数据智能分析平台

Figure 22　Argus: edge-based large-scale video analytics platform

其一, Argus 平台所有模块都按照微服务架构进行了解耦, 因此可以通过容器编排框架将服务无缝扩展到新的计算资源上. 既不需要人工对系统架构进行调整, 也几乎不会增加额外的资源开销. 其二, Argus 系统以边缘视频分析为核心的分布式架构设计, 减少了大量视频流传输带来的网络带宽占用, 避免网络基础设施成为系统规模扩张的瓶颈.

- 普适性指 Argus 作为一个通用视频大数据分析平台, 能够集成不同种类的算法, 适应不同的应用场景. 针对算法普适性问题, Argus 通过良好的模块设计, 支持各层次边缘计算技术通过微服务或视频分析插件形式加入视频流分析流程中. 针对应用场景普适性问题, Argus 在数据源和接入网络方面尽可能支持了多种不同协议和异构设备, 从而能够部署到多种不同场景. 此外, Argus 在视频流分析后的数据分析逻辑, 还可以根据具体工业应用进行定制.

接下来, 本节首先从视频/传感器数据源、接入网络、边缘视频流分析、模块编排和管理、数据存储和挖掘这 5 个主要模块对 Argus 平台顶层架构设计与部分实现进行说明, 然后简要介绍 Argus 平台针对智慧油田项目定制的工业应用.

视频/传感器数据源: 负责采集视频数据并进行预处理, 主要包含视频处理和轻量级模型分析两种功能模块. 其中视频处理模块根据配置好的参数调整视频帧率、比特率, 并对固定摄像头进行背景抠除和 ROI 区域提取. 轻量级模型分析则利用压缩后的轻量级模型以及传统计算机视觉算法对视频流进行分析, 过滤下游分析任务不感兴趣的视频帧并上传部分分析结果. 由于部署用途不同, 同一分析平台覆盖范围内可能有多种异构的终端摄像头. 一方面, 部分摄像头为固定视角, 另一部分摄像头则支持旋转缩放; 另一方面, 不同摄像头计算能力差别极大, 因此支持的预处理任务也不同. 为了减轻异构设备的部署难度, Argus 通过容器方式将预处理模块部署到具有虚拟化能力的对应的摄像头处, 并通过 KubeEdge[2)] 框架根据设备处理能力以及网络状况动态调节预处理模块的参数. 此外, 虽然以视频分析作为核心, Argus 也支持将多种环境和工业传感器的数据收集与上传. 具体来说, 视频流根据预处理后的帧率和比特率通过 RTSP 协议推流到边缘服务器, 视频帧和 ROI 在压缩后通过 UDP 协议上传, 传感器数据和轻量级分析结果则通过 TCP 协议上传.

接入网络: 实现了对多种网络通信协议的网关支持, 一端能够自动连接和配置新增终端设备, 另一端通过光纤直连边缘计算服务器. 大部分摄像头通过 WiFi, 5G 等无线方式接入网络. 少部分工业

2) KubeEdge. KubeEdge: an open platform to enable edge computing. 2021. https://kubeedge.io/en/.

生产相关的传感器则通过蓝牙、Zigbee、ModBus 等多种不同物联网协议连接到接入网关. 由于油田覆盖范围广袤、基础通信设施薄弱, 许多油井附近的关键地点无法被生产网络完全覆盖. 因此, 我们自主研发了高性能自组网 Mesh 节点和路由协议, 能够通过最多 15 跳节点将偏远位置的视频流转发到附近的边缘服务器, 并且支持 500 MB 左右的传输带宽.

边缘视频流分析: 在单个边缘服务器集群上进行负载分发和视频流推理分析, 并将推理结果和资源地址等消息上传到云端. 所有实时视频流或 ROI 会被流调度模块分配到一个或多个视频处理模块中, 传感器数据和初始分析结果则直接被转发到云端进行分析. 大部分视频流会被分配到对应的分析节点, 每个节点的多路视频流按照图 22 所示流程, 经过解码、自适应批处理, 以及目标追踪、多级检测和分类器构成的模型组合推理, 得到每帧视频的分析结果. 视频分析节点基于 NVIDIA 公司的 DeepStream 软件开发工具包[3] 实现, 能够高效利用 GPU 计算资源加速视频解码和模型推理, 还可以按照插件形式添加和连接各个分析子模块, 根据业务需求自定义分析逻辑. 每个视频分析节点构成单独的容器, 并且支持在运行中与外部服务实时交互, 从而实现模型热更新和各插件参数调整. 根据具体业务逻辑, 部分视频流除了实时分析之外还需要进行直播推流. 这部分视频流被同时分配到 PingOS[4] 构成的视频直播与回放模块进行转码和存储, 并将资源播放地址上传到云端.

模块编排和管理: 对系统中所有功能模块进行容器编排和资源管理, 包含了模型仓库、查询优化器和定制化键值存储模块. 其中模型仓库维护了需要使用的深度学习模型的参数、用途, 以及资源占用, 能够根据应用要求和资源状况选择合适的模型部署, 并且支持对模型进行参数量化和校准. 定制化键值存储模块则作为视频分析模块的公共缓存, 存储了部分从视频帧输入到分析结果的映射, 用于同一视频流在不同分析节点执行推理时复用部分结果. 查询优化器与第 4.3.1 小节提到的技术相似, 根据资源限制, 以最大化吞吐量为原则在终端设备和边缘服务器上配置分析任务以及对应的模型. Argus 利用 Kubernetes[5] 和 KubeEdge 框架进行容器编排, 分别将对应的执行计划部署到边缘分析模块和终端视频源节点上, 并根据当前负载情况调节容器数量和资源分配. 例如, 设定视频流的预处理过程, 每个分析节点中的批次大小、追踪器阈值、检测器模型等.

作为整个系统的配置中心, 除了对每个视频流的查询请求配置执行计划之外, 模块编排和管理部分还可以根据网络和计算状况动态调整系统配置. 首先, 在终端视频源处, Argus 根据数据传输的反馈判断网络状况, 以此调整预处理模块的过滤阈值, 确保数据传输不发生拥塞. 其次, Argus 以负载均衡为原则, 动态调整流调度模块, 把具有相同分析请求的视频流在多个相同分析节点间均匀分配. 最后, 在视频流分析节点中, Argus 根据节点返回的分析延迟信息, 动态调整批次大小和追踪、检测间的视频帧数量分配, 避免分析延迟超过阈值.

数据存储和挖掘: 包含对跨摄像头、跨历史的视频分析和传感器元数据的进一步挖掘, 以及对系统日志、运行性能等数据的分析, 并将数据进行持久化存储. 该部分通常部署在云服务器上, 集中接收所有不同位置边缘服务器的分析结果, 其处理能力需要随着边缘节点数量和视频流规模增加而在运行时无缝扩张. 因此, Argus 选取了相应的企业级开源技术方案. 其中, Kafka[6] 是基于订阅 – 发布机制的消息队列系统. 作为系统的总消息代理, 其既负责接收多个边缘服务器上传的实时视频分析元数据供系统处理, 也在其他模块之间提供通信服务, 解耦各个功能模块. Flink[7] 流处理引擎则实时从 Kafka 消息队列中实时拉取大量分析和感知数据, 一方面将元数据进行处理后写入 MySQL [174] 数据库持久

3) Nvidia. NVIDIA deepStream SDK. 2016 https://developer.nvidia.com/deepstream-sdk.
4) PingOStack. PingOS. pingostack, 2021.
5) Kubernetes. Production-grade container orchestration. 2021. https://kubernetes.io/.
6) Kafka. Apache kafka: a distributed streaming platform. 2021. https://kafka.apache.org/.
7) Flink. Apache flink: stateful computations over data streams. 2021. https://flink.apache.org/.

化存储, 另一方面调用 Python 模块进行进一步数据挖掘和分析.

通过对多个视频流的分析数据进行融合, Python 模块实现了跨摄像头行人追踪、人脸识别和权限鉴别等需要跨摄像头或读取数据库的任务, 也可以对历史数据规律进行批量分析. 为了提高分析效率, Argus 利用内存数据库 Redis[8] 对 Python 模块所需的人脸特征等数据进行缓存. 此外, LogStash, ElasticSearch, Kibana 属于 Elastic stack[9] 技术栈的重要组成部分, 专注于日志管理和系统性能监控. 具体来说, LogStash 负责从多个数据源动态采集、解析和转换各种格式的日志及性能指标; ElasticSearch 对 LogStash 导出的数据进行持久化存储, 并构建索引和提供查询; Kibana 则面向用户提供灵活可交互的数据可视化界面.

工业应用: 在 Argus 平台架构的基础上, 根据智慧油田的场景和需求实现了 REST API 和数据可视化, 目前在安防、资产和环境 3 方面提供服务. 在安防监控方面, Argus 实现了入侵报警、车牌统计、人员追踪, 以及特定区域人群聚集行为监控, 涉及目标检测、人脸识别、字符识别和跨摄像头多目标追踪等视频分析任务. 同时, 系统需要根据用户请求进行视频直播与回放. 此外, Argus 还提供了智能视频回放功能, 将多路长时间的视频流根据分析情况融合为一段包含关键信息的摘要, 供安防工作人员快速查找目标. 在资产管理方面, Argus 利用摄像头对生产物资进行智能计数, 并接入油田的企业资源规划系统, 对物资损耗现象进行分析. 在环境检测方面, Argus 通过烟雾识别对火警、油气泄漏情况进行报警.

可以发现, Argus 中视频数据源和接入网络部分运用到了终端设备层次和网络协议相关的边缘视频分析技术. 模块编排和管理部分综合运用了协作层次和边缘/云层次技术. 边缘视频流分析部分则根据配置结果在边缘服务器集群上利用 GPU 资源进行大规模视频分析. 第 4 节提到的边缘实时视频分析领域各层次技术以及未来研究成果, 也可以作为独立容器模块或按照分析算法插件形式无缝嵌入到现有平台中. 例如, 通过对 DeepStream 中的数据源插件按照模型中间层进行修改, 并将数据源处的轻量级模型替换为部分模型, 即可实现设备端和边缘服务器间的纵向模型分割; 查询优化中的各种视频分析流程优化算法也可以直接载入平台中.

作为一个通用分析平台, 除了智慧油田相关任务, Argus 也可以支持交通分析、智慧城市等以大规模监控视频分析为主的业务逻辑. 然而, Argus 主要针对固定摄像头和服务器位置的边缘站点范式, 由云端进行最终的数据分析与存储, 因此对增强现实等面向移动设备且需要低延迟的应用场景较难支持. 我们认为, 面对这类游牧服务场景, 平台还需要在终端设备和边缘服务器之间提供比目前通过容器编排系统间接通信方式更加低延迟的双向通信协议, 并且设计出计算模块和资源状态在不同地理位置的边缘节点间的热切换机制.

6 未来研究方向

本节分别从终端设备、协作和边缘/云这 3 个关键技术层次, 以及实际系统应用角度, 对实时视频分析的边缘计算范式尚待解决的关键问题和未来研究方向进行讨论.

6.1 终端设备层次: 模型性能预测

终端设备层次技术的核心在于将复杂的计算机视觉算法和模型应用到资源有限且架构各异的终端设备上. 不论是采取模型选择、模型压缩, 还是本地缓存或视频帧过滤技术, 系统都需要根据硬件设

8) Redislab. Redis. 2021. https://redis.io/.
9) Elastic. Log monitoring with elasticsearch & elastic stack. 2021. https://www.elastic.co/log-monitoring.

备的计算能力对计算机视觉算法与深度学习模型进行选择, 并且确定分析流程中的参数, 从而权衡推理性能和分析准确度. 其中, 分析准确度仅由模型自身参数决定, 而模型推理性能则受到模型、设备, 以及推理框架等多个方面的共同影响. 现有工作大多基于离线或在线周期性测试来估计当前模型在特定设备上的性能表现 [15, 75, 76, 142]. 这两种方法都需要将每个候选模型在不同硬件设备上进行多次运行后得到平均的推理延迟, 需要较大的计算和时间开销, 仅能应用于候选模型和硬件架构较少的情况. 然而, 随着深度学习技术的发展, 解决同一个问题可以选择的模型数量越来越多, 且同一种模型架构可以衍生出多个版本. 要将如此大量的模型在不同的边缘计算设备上都进行测量, 对于大部分研究者来说难以实现. 因此, 快速预测视频分析模型在边缘计算设备上的性能表现是目前亟需研究的关键问题, 对终端设备层技术具有重要意义.

解决这一问题需要整个研究社区的进一步合作. 短期来看, 可以构建公开的数据库, 由研究人员和硬件厂商共同更新不同模型在不同硬件上的性能表现以及不同数据集上的分析效果. 根据公开数据库, 研究人员和开发者可以一次性或者动态从众多候选中筛选出当前部署环境最适合的算法和配置参数, 减少不同工作对相同模型进行重复测试带来的冗余计算开销. 长期来看, 更重要的是对硬件架构、推理框架、深度学习理论等进行更深入的研究, 快速准确地预测任意的神经网络在特定硬件设备上的推理性能. 特别地, Augur [175] 和 nn-Meter [176] 工作分别基于矩阵乘法和模型内核 (kernel), 对深度神经网络在部分边缘计算设备上的效率进行预测, 具有一定的启示意义.

6.2 协作层次: 网络特性支持

协作层次技术需要实现终端 – 边缘服务器 – 云服务器等不同层级和不同设备间的任务协同. 其中各项优化技术都涉及同层和跨层级边缘计算设备间的网络通信. 随着有线和无线通信技术的发展, 计算机网络基础设施发生着巨大的变化, 诸如 5G, WiFi6, LoRa 和 TSN 等技术带来了不同的网络特性. 我们认为, 对最新的不同网络通信技术的支持和利用, 是协作层次技术需要面对的核心问题.

一方面, 边缘设备在计算和网络通信都具有异构特性, 新的通信技术增加了系统网关实现的复杂程度. 为了实现任务卸载和协作, 边缘实时视频流分析系统需要考虑采用不同网络通信技术的摄像头以及边缘计算设备之间的互联互通. 并且, 对于能量有限的终端设备, 不同通信技术带来的能耗开销同样需要纳入系统考量. 另一方面, 新技术带来的网络特性同样可以优化边缘计算视频分析系统的协作效率. 其中, 5G 网络提供了高可靠性和低延迟 (ultra-reliable and low latency communications, URLLC) 通信服务, 能够确保 1 ms 以内的无线通信延迟, 可以保证部分延迟敏感的分析任务即时卸载到边缘端进行分析. 例如, 支持智能驾驶汽车在高速行驶的同时将任务卸载到边缘服务器并快速收到指令反馈. TSN 则进一步将确定性时延降低到纳秒级别, 可以用于智能汽车内部控制系统的网络通信. 此外, 软件定义网络 (software defined network, SDN) 和网络功能虚拟化 (network function virtualization, NFV) 等技术的推广, 使得系统可以对网络资源进行细粒度切片和分配, 并对视频帧、关键数据、交互信息等不同流量分别进行调度.

6.3 边缘/云层次: 模型持续学习

边缘/云层次技术需要利用边缘和云服务器相对较强的计算能力, 提升视频流分析的吞吐量和准确度. 如本文所调研的, 边缘计算设备计算能力较弱, 大部分需要运行经过定制化或模型压缩的轻量级模型. 这些模型的参数量和网络结构较原始模型简单许多, 因此分析和表达能力相对较弱. 由于视频流中的内容, 例如交通监控摄像头面临的光照条件、人群密度、目标类别分布等, 都可能随着时间发生变化, 定制化模型很难在一次离线训练后同时在多种不同环境下达到理想的分析表现. 因此, 设备

和边缘计算节点上的视频分析模型需要持续学习并适应视频内容的变化, 才能确保对当前内容的分析效果. 仅少部分基于边缘计算的实时视频分析系统考虑到了视频内容的动态性, 大部分都将深度学习模型视作黑盒. 其中有的根据视频内容动态调整分析流程中参数配置 [142] 和模型选择 [73], 还有的根据内容设置视频帧筛选的阈值 [86]. 利用边缘/云服务器上的 GPU 等计算资源, 可以根据视频内容直接自适应地学习模型参数, 能够在复杂环境下更好地提升视频分析系统的准确度.

持续学习方法利用最新的输入周期性地对现有模型进行增量式训练, 是使深度学习模型自动适应视频内容动态变化的有效途径. 在深度学习和计算机视觉领域, 已经存在许多基于持续学习的理论和算法. 例如 iCaRL [177] 基于当前图像流中的已经出现过的类别, 通过样本选择方式增量训练图像分类器的输出类别, JITNet [178] 利用当前视频流输入, 在同一个 GPU 上根据大模型周期性训练小型语义分割模型. 然而, 将持续学习技术应用到基于边缘计算的实时视频流分析系统中仍然面临许多挑战. 若将在线训练和推理都放在边缘服务器上, 可能面临计算资源不足的问题, 导致模型训练影响视频分析任务的实时性. 因此, 需要对模型训练和推理任务进行调度, 甚至进一步考虑在服务多个视频流时选择性地进行模型更新, 在满足延迟、资源开销等前提下最大化持续学习对分析准确度的提升. 若将视频流传输到云端, 由云端进行模型训练再更新边缘端模型, 则面临数据、模型参数持续传输带来的较大带宽占用问题. 因此, 需要通过合理的传输协议, 利用选择性数据上传和参数压缩等策略减少带宽占用, 且不影响在线学习的效果. 部分研究对上述挑战分别进行了初步探索 [179, 180]. 但是更加系统性地、端到端地对在线学习带来的准确度提升与资源消耗进行权衡仍然是一个待解决的难题.

6.4 系统应用: 通用测试平台

基于边缘计算的实时视频流分析具有广泛的应用场景和大量相关研究. 从实际系统应用角度, 该计算范式目前还需要解决系统的可复现性和公平对比问题. 由于边缘计算涉及复杂异构计算设备和多变的网络环境, 且视频分析表现与视频内容关联紧密, 基于不同测试环境和测试数据的工作缺乏公平的横向比较. 随着相关研究的大量涌现, 研究者较难系统地比较不同方法的优劣势和适用场景. 这阻碍了进一步改进和应用. 同时, 由于硬件部署的复杂特性, 即使部分工作公开了源代码和视频数据, 其他研究者也难以对其表现进行复现. 因此, 实现通用的测试平台并提供公开数据集, 有利于后续相关工作的复现、改进和实际应用部署.

测试平台涉及数据集、硬件平台和网络条件 3 个方面工作, 既需要支持边缘实时视频流分析中的常见应用场景, 还需要方便本地复现或远程测试. 第一, 在数据集方面, 边缘实时视频流分析面临第 3 节中提到的多个不同应用场景, 每个应用场景需要不同类型的视频流数据和请求类型, 并且不同请求可能有不同的延迟要求. 公开数据集应该包含监控、交通、增强现实、无人机等有代表性的真实场景视频流. 第二, 在硬件平台方面, 边缘计算场景涉及设备 – 边缘 – 云端 3 个层次的不同硬件, 包括 Raspberry Pi、Nvidia Jetson、智能手机, 甚至 FPGA 等异构设备, 需要通过计算资源仿真或提供远程测试服务的方法降低实验成本. 测试平台还可以将常见的视频分析模块提前进行封装, 并允许研究者以插件形式加入和组合各项技术, 通过容器等技术快速进行硬件或仿真测试. 第三, 在网络方面, 云边端进行任务卸载涉及反复的网络通信, 需要对常见的网络延迟、带宽波动等情况进行记录、模拟与控制, 并支持反复重放实验. 此外, 对于常见的增强现实、大规模交通分析等应用, 还可以基于真实场景和任务设置标准化竞赛, 激励边缘实时视频分析领域从研究走向落地.

7 总结

实时视频流分析是边缘计算最成功的应用场景; 边缘计算范式也是实时视频分析能成功部署的重要支撑. 本文对近年来针对实时视频流分析的边缘计算系统进行了归纳和综述. 首先, 本文分别介绍了视频分析和边缘计算涉及的背景知识, 并分析了边缘实时视频流分析系统的优化目标和其面临的关键挑战. 接着, 本文针对这些挑战, 从终端设备层次、云边端跨层多设备协作和边缘/云服务层次阐述了各项工作中涉及的关键技术与效果. 基于对各项核心技术的理解和整合, 本文提出了基于边缘计算的智能大数据分析平台 Argus, 并对其顶层设计与部分实现细节进行了介绍. 目前, 针对实时视频流分析的边缘计算范式依然面临着一些尚待解决的难题, 本文最后对该领域的未来研究方向进行了讨论.

参考文献

1 Cisco. Cisco Annual Internet Report (2018–2023) White Paper. 2020

2 Liu H H, Chen S Y, Kubota N. Intelligent video systems and analytics: a survey. IEEE Trans Ind Inf, 2013, 9: 1222–1233

3 Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS), Lake Tahoe, 2011. 1–9

4 Karpathy A, Toderici G, Shetty S, et al. Large-scale video classification with convolutional neural networks. In: Proceedings of 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, 2014. 1725–1732

5 Olatunji I E, Cheng C H. Video analytics for visual surveillance and applications: an overview and survey. In: Machine Learning Paradigms: Applications of Learning and Analytics in Intelligent Systems, Learning and Analytics in Intelligent Systems. Cham: Springer International Publishing, 2019. 475–515

6 Mach P, Becvar Z. Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun Surv Tut, 2017, 19: 1628–1656

7 Chen J, Ran X. Deep learning with edge computing: a review. Proc IEEE, 2019, 107: 1655–1674

8 Liu L, Li H, Gruteser M. Edge assisted real-time object detection for mobile augmented reality. In: Proceedings of the 25th Annual International Conference on Mobile Computing and Networking (MobiCom), New York, 2019. 1–16

9 Kang D, Emmons J, Abuzaid F, et al. NoScope: optimizing neural network queries over video at scale. Proc VLDB Endow, 2017, 10: 1586–1597

10 Hsieh K, Ananthanarayanan G, Bodik P, et al. Focus: querying large video datasets with low latency and low cost. In: Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI), Carlsbad, 2018. 269–286

11 Kang D, Bailis P, Zaharia M. BlazeIt: optimizing declarative aggregation and limit queries for neural network-based video analytics. Proc VLDB Endow, 2019, 13: 533–546

12 Anderson M R, Cafarella M, Ros G, et al. Physical representation-based predicate optimization for a visual analytics database. In: Proceedings of IEEE 35th International Conference on Data Engineering (ICDE), Macao, 2019. 1466–1477

13 Lai Z, Han C, Liu C, et al. Finding interesting frames in deep video analytics: a top-K approach. 2020. ArXiv:2003.00773

14 Ananthanarayanan G, Bahl P, Bodik P, et al. Real-time video analytics: the killer app for edge computing. Computer, 2017, 50: 58–67

15 Zhang H, Ananthanarayanan G, Bodik P, et al. Live video analytics at scale with approximation and delay-tolerance. In: Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2017.

377–392

16 Jain S, Nguyen V, Gruteser M, et al. Panoptes: servicing multiple applications simultaneously using steerable cameras. In: Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Pittsburgh, 2017. 119–130

17 Hung C C, Ananthanarayanan G, Bodik P, et al. VideoEdge: processing camera streams using hierarchical clusters. In: Proceedings of IEEE/ACM Symposium on Edge Computing, Seattle, 2018. 115–131

18 Mao Y, You C, Zhang J, et al. A survey on mobile edge computing: the communication perspective. IEEE Commun Surv Tut, 2017, 19: 2322–2358

19 Li C, Xue Y, Wang J, et al. Edge-oriented computing paradigms. ACM Comput Surv, 2018, 51: 1–34

20 Zhou Z, Chen X, Li E, et al. Edge intelligence: paving the last mile of artificial intelligence with edge computing. Proc IEEE, 2019, 107: 1738–1762

21 Murshed M G S, Murphy C, Hou D, et al. Machine learning at the network edge: a survey. 2020. ArXiv:1908.00080

22 Zhang Q, Sun H, Wu X, et al. Edge video analytics for public safety: a review. Proc IEEE, 2019, 107: 1675–1696

23 Jedari B, Premsankar G, Illahi G, et al. Video caching, analytics, and delivery at the wireless edge: a survey and future directions. IEEE Commun Surv Tut, 2021, 23: 431–471

24 Wiegand T, Sullivan G J, Bjontegaard G, et al. Overview of the H.264/AVC video coding standard. IEEE Trans Circ Syst Video Technol, 2003, 13: 560–576

25 Sullivan G J, Ohm J R, Han W J, et al. Overview of the high efficiency video coding (HEVC) standard. IEEE Trans Circ Syst Video Technol, 2012, 22: 1649–1668

26 Yi J, Kim S, Kim J, et al. Supremo: cloud-assisted low-latency super-resolution in mobile devices. IEEE Trans Mobile Comput, 2020. doi: 10.1109/TMC.2020.3025300

27 Benezeth Y, Jodoin P M, Emile B, et al. Comparative study of background subtraction algorithms. J Electron Imag, 2010, 19: 033003

28 Behbahani S, Asadi S, Ashtiyani M, et al. Analysing optical flow based methods. In: Proceedings of IEEE International Symposium on Signal Processing and Information Technology, Giza, 2007. 133–137

29 Viola P, Jones M. Rapid object detection using a boosted cascade of simple features. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Kauai, 2001

30 Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell, 2017, 39: 1137–1149

31 Redmon J, Divvala S, Girshick R, et al. You only look once: unified, real-time object detection. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 2016. 779–788

32 Zhao Z Q, Zheng P, Xu S T, et al. Object detection with deep learning: a review. IEEE Trans Neural Netw Learn Syst, 2019, 30: 3212–3232

33 Kortli Y, Jridi M, Falou A A, et al. Face recognition systems: a survey. Sensors, 2020, 20: 342

34 Yilmaz A, Javed O, Shah M. Object tracking: a survey. ACM Comput Surv, 2006, 38: 13

35 Bertinetto L, Valmadre J, Henriques J F, et al. Fully-convolutional siamese networks for object tracking. In: Proceedings of the 14th European Conference on Computer Vision Workshop, Amsterdam, 2016. 850–865

36 Bewley A, Ge Z, Ott L, et al. Simple online and realtime tracking. In: Proceedings of IEEE International Conference on Image Processing (ICIP), Phoenix, 2016. 3464–3468

37 Zhang S, Zhu Y, Roy-Chowdhury A. Tracking multiple interacting targets in a camera network. Comput Vis Image Und, 2015, 134: 64–73

38 Liu W, Anguelov D, Erhan D, et al. SSD: single shot multiBox detector. In: Proceedings of the 14th European Conference on Computer Vision, Amsterdam, 2016. 21–37

39 Zhu X, Xiong Y, Dai J, et al. Deep feature flow for video recognition. In: Proceedings of IEEE Conference on

Computer Vision and Pattern Recognition (CVPR), Honolulu, 2017. 4141–4150

40 Feichtenhofer C, Pinz A, Zisserman A. Detect to track and track to detect. In: Proceedings of IEEE International Conference on Computer Vision (ICCV), Venice, 2017. 3057–3065

41 Yi S, Hao Z, Qin Z, et al. Fog computing: platform and applications. In: Proceedings of the 3rd IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), Washington, 2015. 73–78

42 Satyanarayanan M. Pervasive computing: vision and challenges. IEEE Pers Commun, 2001, 8: 10–17

43 Noghabi S A, Cox L, Agarwal S, et al. The emerging landscape of edge computing. GetMobile-Mobile Comp Comm, 2020, 23: 11–20

44 Zhang T, Chowdhery A, Bahl V, et al. The design and implementation of a wireless video surveillance system. In: Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom), New York, 2015. 426–438

45 Hu P, Ning H, Qiu T, et al. Fog computing based face identification and resolution scheme in Internet of Things. IEEE Trans Ind Inf, 2017, 13: 1910–1920

46 Zhang Q, Zhang Q, Shi W, et al. Distributed collaborative execution on the edges and its application to AMBER alerts. IEEE Internet Things J, 2018, 5: 3580–3593

47 Barthélemy J, Verstaevel N, Forehead H, et al. Edge-computing video analytics for real-time traffic monitoring in a smart city. Sensors, 2019, 19: 2048

48 Kar G, Jain S, Gruteser M, et al. Real-time traffic estimation at vehicular edge nodes. In: Proceedings of the 2nd ACM/IEEE Symposium on Edge Computing, New York, 2017. 1–13

49 Qiu H, Liu X, Rallapalli S, et al. Kestrel: video analytics for augmented multi-camera vehicle tracking. In: Proceedings of IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI), Orlando, 2018. 48–59

50 Jain P, Manweiler J, Choudhury R R. OverLay: practical mobile augmented reality. In: Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services, (MobiSys), New York, 2015. 331–344

51 Al-Shuwaili A, Simeone O. Energy-efficient resource allocation for mobile edge computing-based augmented reality applications. IEEE Wireless Commun Lett, 2017, 6: 398–401

52 Ran X, Chen H, Zhu X, et al. DeepDecision: a mobile deep learning framework for edge video analytics. In: Proceedings of IEEE Conference on Computer Communications (INFOCOM), Honolulu, 2018. 1421–1429

53 Chen T Y H, Ravindranath L, Deng S, et al. Glimpse: continuous, real-time object recognition on mobile devices. In: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys), Seoul, 2015. 155–168

54 Merino L, Caballero F, Martínez-de-Dios J R, et al. An unmanned aircraft system for automatic forest fire monitoring and measurement. J Intell Robot Syst, 2012, 65: 533–548

55 Motlagh N H, Bagaa M, Taleb T. UAV-based IoT platform: a crowd surveillance use case. IEEE Commun Mag, 2017, 55: 128–134

56 Kalatzis N, Avgeris M, Dechouniotis D, et al. Edge computing in IoT ecosystems for UAV-enabled early fire detection. In: Proceeding of IEEE International Conference on Smart Computing (SMARTCOMP), Taormina, 2018. 106–114

57 Ha K, Chen Z, Hu W, et al. Towards wearable cognitive assistance. In: Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys), New York, 2014. 68–81

58 Zhao S, Wang J, Leng H, et al. Edge-based Wearable Systems for Cognitive Assistance: Design Challenges, Solution Framework, and Application to Emergency Healthcare. Technical Report. Pittsburgh: Carnegie Mellon University, 2020

59 Wang J, Feng Z, George S, et al. Towards scalable edge-native applications. In: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, New York, 2019. 152–165

60 Han S, Pool J, Tran J, et al. Learning both weights and connections for efficient neural networks. In: Proceedings of

the 28th International Conference on Neural Information Processing Systems (NeruIPS), Cambridge, 2015. 1135–1143

61 Han S, Mao H, Dally W J. Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding. In: Proceedings of the 4th International Conference on Learning Representations (ICLR), San Juan, 2016. 1–14

62 Yang T, Chen Y, Sze V. Designing energy-efficient convolutional neural networks using energy-aware pruning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, 2017. 6071–6079

63 Mao J, Yang Z, Wen W, et al. MeDNN: a distributed mobile system with enhanced partition and deployment for large-scale DNNs. In: Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, 2017. 751–756

64 He Y, Lin J, Liu Z, et al. AMC: AutoML for model compression and acceleration on mobile devices. In: Proceedings of the European Conference on Computer Vision (ECCV), Munich, 2018. 784–800

65 Lane N D, Bhattacharya S, Georgiev P, et al. DeepX: a software accelerator for low-power deep learning inference on mobile devices. In: Proceedings of 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Vienna, 2016. 1–12

66 Mathur A, Lane N D, Bhattacharya S, et al. DeepEye: resource efficient local execution of multiple deep vision models using wearable commodity hardware. In: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys), New York, 2017. 68–81

67 Bhattacharya S, Lane N D. Sparsification and separation of deep learning layers for constrained resource inference on wearables. In: Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM (SenSys), New York, 2016. 176–189

68 Huynh L N, Lee Y, Balan R K. DeepMon: mobile GPU-based deep learning framework for continuous vision applications. In: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys), New York, 2017. 82–95

69 Howard A G, Zhu M, Chen B, et al. MobileNets: efficient convolutional neural networks for mobile vision applications. 2017. ArXiv:1704.04861

70 Zhang X, Zhou X, Lin M, et al. ShuffleNet: an extremely efficient convolutional neural network for mobile devices. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, 2018. 6848–6856

71 Fang B, Zeng X, Zhang M. NestDNN: resource-aware multi-tenant on-device deep learning for continuous mobile vision. In: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom), New York, 2018. 115–127

72 Tan M, Chen B, Pang R, et al. MnasNet: platform-aware neural architecture search for mobile. 2019. ArXiv:1807.11626

73 Taylor B, Marco V S, Wolff W, et al. Adaptive deep learning model selection on embedded systems. In: Proceedings of the 19th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems, New York, 2018. 31–43

74 Xu R, Zhang C L, Wang P C, et al. ApproxDet: content and contention-aware approximate object detection for mobiles. In: Proceedings of the 18th Conference on Embedded Networked Sensor Systems (SenSys), New York, 2020. 449–462

75 Liu S, Lin Y, Zhou Z, et al. On-demand deep model compression for mobile devices: a usage-driven model selection framework. In: Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys), New York, 2018. 389–400

76 Han S, Shen H, Philipose M, et al. MCDNN: an approximation-based execution framework for deep stream pro-

cessing under resource constraints. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys), New York, 2016. 123–136

77 Xu W, Zheng Y, Jiahang W. EdgeDuet: tiling small object detection for edge assisted autonomous mobile vision. In: Proceedings of IEEE Conference on Computer Communications (INFOCOM), Vancouver, 2021. 1–10

78 Henriques J F, Caseiro R, Martins P, et al. High-speed tracking with kernelized correlation filters. IEEE Trans Pattern Anal Mach Intell, 2015, 37: 583–596

79 Zhang W, Han B, Hui P. Jaguar: low latency mobile augmented reality with flexible tracking. In: Proceedings of the 26th ACM International Conference on Multimedia, New York, 2018. 355–363

80 Wu H, Feng J, Tian X, et al. EMO: real-time emotion recognition from single-eye images for resource-constrained eyewear devices. In: Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (MobiSys), New York, 2020. 448–461

81 LiKamWa R, Zhong L. Starfish: efficient concurrency support for computer vision applications. In: Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys), New York, 2015. 213–226

82 Guo P, Hu W. Potluck: cross-application approximate deduplication for computation-intensive mobile applications. In: Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems, New York, 2018. 271–284

83 Cavigelli L, Degen P, Benini L. CBinfer: change-based inference for convolutional neural vetworks on video data. In: Proceedings of the 11th International Conference on Distributed Smart Cameras (ICDSC), New York, 2017. 1–8

84 Xu M, Zhu M, Liu Y, et al. DeepCache: principled cache for mobile deep vision. In: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom), New York, 2018. 129–144

85 Gammeter S, Gassmann A, Bossard L, et al. Server-side object recognition and client-side object tracking for mobile augmented reality. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), San Francisco, 2010. 1–8

86 Li Y, Padmanabhan A, Zhao P, et al. Reducto: on-camera filtering for resource-efficient real-time video analytics. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM), Virtual Event, 2020. 359–376

87 Luo H, Xie W, Wang X, et al. Detect or track: towards cost-effective video object detection/tracking. In: Proceedings of AAAI Conference on Artificial Intelligence, Honolulu, 2019. 8803–8810

88 Canel C, Kim T, Zhou G, et al. Scaling video analytics on constrained edge nodes. In: Proceedings of the 2nd SysML Conference, Palo Alto, 2019. 1–12

89 Zhang C, Cao Q, Jiang H, et al. FFS-VA: a fast filtering system for large-scale video analytics. In: Proceedings of the 47th International Conference on Parallel Processing, New York, 2018. 1–10

90 Apicharttrisorn K, Ran X, Chen J, et al. Frugal following: power thrifty object detection and tracking for mobile augmented reality. In: Proceedings of the 17th Conference on Embedded Networked Sensor Systems (SenSys), New York, 2019. 96–109

91 Wang J, Feng Z, Chen Z, et al. Bandwidth-efficient live video analytics for drones via edge computing. In: Proceedings of IEEE/ACM Symposium on Edge Computing, Bellevue, 2018. 159–173

92 Jain S, Zhang X, Zhou Y, et al. Spatula: efficient cross-camera video analytics on large camera networks. In: Proceedings of IEEE/ACM Symposium on Edge Computing, San Jose, 2020. 110–124

93 Jang S Y, Lee Y, Shin B, et al. Application-aware IoT camera virtualization for video analytics edge computing. In: Proceedings of 2018 IEEE/ACM Symposium on Edge Computing, Bellevue, 2018. 132–144

94 Ding H, Guo Y, Li X, et al. Beef up the edge: spectrum-aware placement of edge computing services for the Internet

of Things. IEEE Trans Mobile Comput, 2019, 18: 2783–2795

95 Wang H, Xie J. User preference based energy-aware mobile AR system with edge computing. In: Proceedings of IEEE Conference on Computer Communications (INFOCOM), Virtual Event, 2020. 1379–1388

96 Wang C, Zhang S, Chen Y, et al. Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics. In: Proceedings of IEEE Conference on Computer Communications (INFOCOM), Toronto, 2020. 257–266

97 Liu Q, Huang S, Opadere J, et al. An edge network orchestrator for mobile augmented reality. In: Proceedings of IEEE Conference on Computer Communications (INFOCOM), Honolulu, 2018. 756–764

98 Trinh H, Calyam P, Chemodanov D, et al. Energy-aware mobile edge computing and routing for low-latency visual data processing. IEEE Trans Multimedia, 2018, 20: 2562–2577

99 Zhang W, He Z, Liu L, et al. Elf: accelerate high-resolution mobile deep vision with content-aware distributed offloading. In: Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MobiCom), New Orleans, 2021

100 Kang Y, Hauswald J, Gao C, et al. Neurosurgeon: collaborative intelligence between the cloud and mobile edge. SIGARCH Comput Archit News, 2017, 45: 615–629

101 Li H, Ota K, Dong M. Learning IoT in edge: deep learning for the Internet of Things with edge computing. IEEE Network, 2018, 32: 96–101

102 Teerapittayanon S, McDanel B, Kung H T. BranchyNet: fast inference via early exiting from deep neural networks. In: Proceedings of the 23rd International Conference on Pattern Recognition (ICPR), Cancun, 2016. 2464–2469

103 Teerapittayanon S, McDanel B, Kung H T. Distributed deep neural networks over the cloud, the edge and end devices. In: Proceedings of IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, 2017. 328–339

104 Li E, Zhou Z, Chen X. Edge intelligence: on-demand deep learning model co-inference with device-edge synergy. In: Proceedings of Workshop on Mobile Edge Communications, New York, 2018. 31–36

105 Laskaridis S, Venieris S I, Almeida M, et al. SPINN: synergistic progressive inference of neural networks over device and cloud. In: Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom), New York, 2020. 1–15

106 Ren P, Qiao X, Huang Y, et al. Edge-assisted distributed DNN collaborative computing approach for mobile web augmented reality in 5G networks. IEEE Network, 2020, 34: 254–261

107 Jeong H J, Lee H J, Shin C H, et al. IONN: incremental offloading of neural network computations from mobile devices to edge servers. In: Proceedings of ACM Symposium on Cloud Computing, New York, 2018. 401–411

108 Huang J, Samplawski C, Ganesan D, et al. CLIO: enabling automatic compilation of deep learning pipelines across IoT and cloud. In: Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom), New York, 2020. 1–12

109 Hu C, Bao W, Wang D, et al. Dynamic adaptive DNN surgery for inference acceleration on the edge. In: Proceedings of IEEE Conference on Computer Communications (INFOCOM), Paris, 2019. 1423–1431

110 Xu M, Qian F, Zhu M, et al. DeepWear: adaptive local offloading for on-wearable deep learning. IEEE Trans Mobile Comput, 2020, 19: 314–330

111 Wang L, Xiang L, Xu J, et al. Context-aware deep model compression for edge cloud computing. In: Proceedings of IEEE 40th International Conference on Distributed Computing Systems (ICDCS), Singapore, 2020. 787–797

112 Yi S, Hao Z, Zhang Q, et al. LAVEA: latency-aware video analytics on edge computing platform. In: Proceedings of the 2nd ACM/IEEE Symposium on Edge Computing, New York, 2017. 1–13

113 Wang S, Yang S, Zhao C. SurveilEdge: real-time video query based on collaborative cloud-edge deep learning. In: Proceedings of IEEE Conference on Computer Communications (INFOCOM), Toronto, 2020. 2519–2528

114 Long C, Cao Y, Jiang T, et al. Edge computing framework for cooperative video processing in multimedia IoT systems. IEEE Trans Multimedia, 2018, 20: 1126–1139

115 Jiang J, Zhou Y, Ananthanarayanan G, et al. Networked cameras are the new big data clusters. In: Proceedings of Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo), Los Cabos, 2019. 1–7

116 Zeng X, Fang B, Shen H, et al. Distream: scaling live video analytics with workload-adaptive distributed edge intelligence. In: Proceedings of the 18th Conference on Embedded Networked Sensor Systems (SenSys), New York, 2020. 409–421

117 Mao J, Chen X, Nixon K W, et al. MoDNN: local distributed mobile computing system for deep neural network. In: Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, 2017. 1396–1401

118 Zhao Z, Barijough K M, Gerstlauer A. DeepThings: distributed adaptive deep learning inference on resource-constrained IoT edge clusters. IEEE Trans Comput-Aided Des Integr Circ Syst, 2018, 37: 2348–2359

119 Zhou L, Samavatian M H, Bacha A, et al. Adaptive parallel execution of deep neural networks on heterogeneous edge devices. In: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, New York, 2019. 195–208

120 Zhang B, Jin X, Ratnasamy S, et al. AWStream: adaptive wide-area streaming analytics. In: Proceedings of Conference of the ACM Special Interest Group on Data Communication, Budapest, 2018. 236–252

121 Rabkin A, Arye M, Sen S, et al. Aggregation and degradation in JetStream: streaming analytics in the wide area. In: Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI), Seattle, 2014. 275–288

122 Wang Y, Wang W, Zhang J, et al. Bridging the edge-cloud barrier for real-time advanced vision analytics. In: Proceedings of the 11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud), Renton, 2019. 1–7

123 Wang X, Chowdhery A, Chiang M. SkyEyes: adaptive video streaming from UAVs. In: Proceedings of the 3rd Workshop on Hot Topics in Wireless, New York, 2016. 2–6

124 Pakha C, Chowdhery A, Jiang J. Reinventing video streaming for distributed vision analytics. In: Proceedings of the 10th USENIX Conference on Hot Topics in Cloud Computing (HotCloud), 2018. 1–7

125 Du K, Pervaiz A, Yuan X, et al. Server-driven video streaming for deep learning inference. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM), Virtual Event, 2020. 557–570

126 Yi J, Choi S, Lee Y. EagleEye: wearable camera-based person identification in crowded urban spaces. In: Proceedings of the 26th Annual International Conference on Mobile Computing and Networking (MobiCom), New York, 2020. 1–14

127 Ko J H, Na T, Amir M F, et al. Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained Internet-of-Things platforms. In: Proceedings of the 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, 2018. 1–6

128 Li H, Hu C, Jiang J, et al. JALAD: joint accuracy- and latency-aware deep structure decoupling for edge-cloud execution. In: Proceedings of IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), Singapore, 2018. 671–678

129 Emmons J, Fouladi S, Ananthanarayanan G, et al. Cracking open the DNN black-box: video analytics with DNNs across the camera-cloud boundary. In: Proceedings of the Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo), New York, 2019. 27–32

130 Matsubara Y, Baidya S, Callegaro D, et al. Distilled split deep neural networks for edge-assisted real-time systems. In: Proceedings of Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo), New York, 2019. 21–26

131 Hu D, Krishnamachari B. Fast and accurate streaming CNN inference via communication compression on the edge. In: Proceedings of IEEE/ACM 5th International Conference on Internet-of-Things Design and Implementation (IoTDI), Sydney, 2020. 157–163

132 Simoens P, Xiao Y, Pillai P, et al. Scalable crowd-sourcing of video from mobile devices. In: Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, New York, 2013. 139–152

133 Wang J, Amos B, Das A, et al. A scalable and privacy-aware IoT service for live video analytics. In: Proceedings of the 8th ACM on Multimedia Systems Conference, New York, 2017. 38–49

134 Hassan E T, Hasan R, Shaffer P, et al. Cartooning for enhanced privacy in lifelogging and streaming videos. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, 2017. 1333–1342

135 Gafni O, Wolf L, Taigman Y. Live face de-identification in video. In: Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, 2019. 9377–9386

136 Bertran M, Martinez N, Papadaki A, et al. Adversarially learned representations for information obfuscation and inference. In: Proceedings of the 36th International Conference on Machine Learning (ICML), Long Beach, 2019. 614–623

137 Raval N, Machanavajjhala A, Cox L P. Protecting visual secrets using adversarial nets. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, 2017. 25–28

138 Wang J, Zhang J, Bao W, et al. Not just privacy: improving performance of private deep learning in mobile cloud. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, New York, 2018. 2407–2416

139 Wu H, Tian X, Li M, et al. PECAM: privacy-enhanced video streaming and analytics via securely-reversible transformation. In: Proceedings of the 27th Annual International Conference On Mobile Computing and Networking (MobiCom), New Orleans, 2021

140 Chaiken R, Jenkins B, Larson P, et al. SCOPE: easy and efficient parallel processing of massive data sets. Proc VLDB Endow, 2008, 1: 1265–1276

141 Lu Y, Chowdhery A, Kandula S. Optasia: a relational platform for efficient large-scale video analytics. In: Proceedings of ACM Symposium on Cloud Computing (SoCC), New York, 2016. 57–70

142 Jiang J, Ananthanarayanan G, Bodik P, et al. Chameleon: scalable adaptation of video analytics. In: Proceedings of Conference of the ACM Special Interest Group on Data Communication (SIGCOMM), Budapest, 2018. 253–266

143 Zhang W, Li S, Liu L, et al. Hetero-edge: orchestration of real-time vision applications on heterogeneous edge clouds. In: Proceedings of IEEE Conference on Computer Communications (INFOCOM), Paris, 2019. 1270–1278

144 Olston C, Fiedel N, Gorovoy K, et al. TensorFlow-serving: flexible, high-performance ML serving. 2017. ArXiv:1712.06139

145 Crankshaw D, Wang X, Zhou G, et al. Clipper: a low-latency online prediction serving system. In: Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2017. 613–627

146 Narra K G, Lin Z, Ananthanarayanan G, et al. Collage inference: using coded redundancy for low variance distributed image classification. 2019. ArXiv:1904.12222

147 Shen H, Chen L, Jin Y, et al. Nexus: a GPU cluster engine for accelerating DNN-based video analysis. In: Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP), New York, 2019. 322–337

148 Crankshaw D, Sela G E, Mo X, et al. InferLine: latency-aware provisioning and scaling for prediction serving pipelines. In: Proceedings of ACM Symposium on Cloud Computing, Virtual Event, 2020. 477–491

149 Zhang D, Vance N, Zhang Y, et al. EdgeBatch: towards AI-empowered optimal task batching in intelligent edge systems. In: Proceedings of IEEE Real-Time Systems Symposium (RTSS), Virtual Event, 2019. 366–379

150 Fang Z, Lin J H, Srivastava M B, et al. Multi-tenant mobile offloading systems for real-time computer vision

applications. In: Proceedings of the 20th International Conference on Distributed Computing and Networking (ICDCN), New York, 2019. 21–30

151 Yang Z, Nahrstedt K, Guo H, et al. DeepRT: a soft real time scheduler for computer vision applications on the edge. In: Proceedings of IEEE/ACM Symposium on Edge Computing, San Jose, 2021. 1–14

152 Dhakal A, Kulkarni S G, Ramakrishnan K K. ECML: improving efficiency of machine learning in edge clouds. In: Proceedings of IEEE 9th International Conference on Cloud Networking (CloudNet), Virtual Event, 2020. 1–6

153 Jiang A H, Wong D L K, Canel C, et al. Mainstream: dynamic stem-sharing for multi-tenant video processing. In: Proceedings of 2018 USENIX Annual Technical Conference (ATC), 2018. 29–42

154 Narayanan D, Santhanam K, Phanishayee A, et al. Accelerating deep learning workloads through efficient multi-model execution. In: Proceedings of Advances in Neural Information Processing Systems (NeuraIPS) Workshop, Montreal, 2018. 1–8

155 Jeong J S, Kim S, Yu G I, et al. Accelerating multi-model inference by merging DNNs of different weights. 2020. ArXiv:2009.13062

156 Venugopal S, Gazzetti M, Gkoufas Y, et al. Shadow puppets: cloud-level accurate AI inference at the speed and economy of edge. In: Proceedings of USENIX Workshop on Hot Topics in Edge Computing (HotEdge), Boston, 2018

157 Drolia U, Guo K, Tan J, et al. Cachier: edge-caching for recognition applications. In: Proceedings of IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, 2017. 276–286

158 Drolia U, Guo K, Narasimhan P. Precog: prefetching for image recognition applications at the edge. In: Proceedings of the 2nd ACM/IEEE Symposium on Edge Computing, New York, 2017. 1–13

159 Guo P, Hu B, Li R, et al. FoggyCache: cross-device approximate computation reuse. In: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom), New York, 2018. 19–34

160 Liu Z, Lan G, Stojkovic J, et al. CollabAR: edge-assisted collaborative image recognition for mobile augmented reality. In: Proceedings of the 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Sydney, 2020. 301–312

161 Younis A, Qiu B, Pompili D. Latency-aware hybrid edge cloud framework for mobile augmented reality applications. In: Proceedings of the 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), Como, 2020. 1–9

162 Russakovsky O, Deng J, Su H, et al. ImageNet large scale visual recognition challenge. Int J Comput Vis, 2015, 115: 211–252

163 Deng B L, Li G, Han S, et al. Model compression and hardware acceleration for neural networks: a comprehensive survey. Proc IEEE, 2020, 108: 485–532

164 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, 2016. 770–778

165 Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-ResNet and the impact of residual connections on learning. In: Proceedings of AAAI Conference on Artificial Intelligence, San Francisco, 2017. 4278–4284

166 Szegedy C, Liu W, Jia Y Q, et al. Going deeper with convolutions. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, 2015. 1–9

167 Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Commun ACM, 2008, 51: 107–113

168 Stockhammer T. Dynamic adaptive streaming over HTTP: standards and design principles. In: Proceedings of the 2nd Annual ACM Conference on Multimedia Systems, New York, 2011. 133–144

169 Huang G, Liu Z, van der Maaten L, et al. Densely connected convolutional networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, 2017. 2261–2269

170 Zhu J Y, Park T, Isola P, et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of IEEE International Conference on Computer Vision (ICCV), Venice, 2017. 2242–2251

171 Zhou J, Bruno N, Wu M C, et al. SCOPE: parallel databases meet MapReduce. VLDB J, 2012, 21: 611–636
172 Dhakal A, Kulkarni S G, Ramakrishnan K K. GSLICE: controlled spatial sharing of GPUs for a scalable inference platform. In: Proceedings of the 11th ACM Symposium on Cloud Computing (SoCC), Virtual Event, 2020. 492–506
173 Rublee E, Rabaud V, Konolige K, et al. ORB: an efficient alternative to SIFT or SURF. In: Proceedings of 2011 International Conference on Computer Vision (ICCV), Barcelona, 2011. 2564–2571
174 Widenius M, Axmark D, DuBois P. MySQL Reference Manual. Sebastopol: O'Reilly & Associates, Inc., 2002. 1–566
175 Lu Z, Rallapalli S, Chan K, et al. Augur: modeling the resource requirements of ConvNets on mobile devices. IEEE Trans Mobile Comput, 2021, 20: 352–365
176 Zhang L L, Han S, Wei J, et al. nn-Meter: towards accurate latency prediction of deep-learning model inference on diverse edge devices. In: Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys), New York, 2021. 81–93
177 Rebuffi S A, Kolesnikov A, Sperl G, et al. iCaRL: incremental classifier and representation learning. In: Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, 2017. 5533–5542
178 Mullapudi R T, Chen S, Zhang K, et al. Online model distillation for efficient video inference. In: Proceedings of 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, 2019. 3572–3581
179 Bhardwaj R, Xia Z, Ananthanarayanan G, et al. Ekya: continuous learning of video analytics models on edge compute servers. 2020. ArXiv:2012.10557
180 Khani M, Hamadanian P, Nasr-Esfahany A, et al. Real-time video inference on edge devices via adaptive model streaming. 2020. ArXiv:2006.06628

Edge computing technologies for streaming video analytics

Zheng YANG*, Xiaowu HE, Jiahang WU, Xu WANG & Yi ZHAO

School of Software, Tsinghua University, Beijing 100084*, China*
* Corresponding author. E-mail: yangzheng@tsinghua.edu.cn

Abstract Real-time streaming video analytics is important in applications such as intelligent surveillance, smart city, and autonomous driving. However, large-scale streaming video analytics is impractical on the cloud, due to its high computation demand, large bandwidth consumption and stringent latency requirement. The emerging edge computing paradigm can effectively solve these problems by pushing computation from the cloud to devices and servers at the network edge. To this end, this article conducts a comprehensive survey on edge computing technologies for real-time streaming video analytics. Firstly, it introduces the background of video analytics and edge computing, as well as the typical application of edge-based steaming video analytics. Then, it proposes the performance indicators and challenges faced by the existing systems. Afterwards, the key technologies in this field are introduced in detail from the device level, collaboration level, and edge/cloud server level, including model compression and selection, local caching, frame filtering, task offloading, streaming protocol, privacy protection, query optimization, inference acceleration, and edge caching. Based on the integration of above core technologies, this article proposes an edge-based large-scale video analytics platform, Argus, which provides systematic support for the real-time video stream analytics on video collection, model inference, data mining, and log management. Argus has been successfully deployed in the smart oilfield scenario. Last but not least, this article discusses the open issues on edge-based streaming video analytics, in the hope of inspiring future research ideas.

Keywords edge computing, video analytics, model compression, task offloading, query optimization